NRL/93-2172/94-05

Rapid Prototyping of Application Specific Signal Processors (RASSP)

First Interim Technical Status Report
12 September 1994

Data Item A002

Prepared For
Advanced Research Projects Agency
Naval Research Laboratory

LOCKHEED SANDERS, INC.
P.O. Box 868 - Nashua, NH 03061-0868

AVY-L-S-00108-101-A

NRL/93-2172/94-05

Rapid Prototyping of Application Specific Signal Processors (RASSP)

First Interim Technical Status Report
23 December 1994

Data Item A002

Contract N00014-93-C-2172

Prepared By

Lockheed Sanders, Inc.
Hughes Aircraft
Motorola
ISX Corporation

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

Prepared For

Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, Virginia 22203-1714

Naval Research Laboratory
4555 Overlook Avenue
Washington, DC 20357-5347

LOCKHEED SANDERS, INC.
P.O. Box 868 - Nashua, NH 03061-0868

AVY-L-S-00108-101-A

ABSTRACT

First Interim Technical Status Report for the RASSP Program covering the period 1 August 1993 through 31 July 1994. Principal Program achievements are described for the RASSP System Engineering, RASSP Design Environment, RASSP Demonstrations and RASSP Process Proliferation Teams, as well as Technical Review Board transactions.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST IF ILLUSTRATIONS

LIST OF TABLES

ACRONYMS AND ABBREVIATIONS

The following acronyms and abbreviations are used in the current report. A comprehensive list of RASSP acronyms and abbreviations is being developed.

ASSET	A Source for Software Engineering Technology
CFI	Computer Aided Design Framework Initiative
DSP	Digital Signal Processing
IRST	Infrared Search and Track
MIMD	Multiple Instruction Multiple Data
RASSP	Rapid Prototyping of Application Specific Signal Processors
RDE	RASSP Design Environment
RSE	RASSP System Engineering
SCI	Scaleable Coherent Interface
SEER	System Evaluation and Estimation of Resources
STARS	Software Technology for Adaptable, Reliable Systems
VHDL	Very High Speed Integrated Circuit High Level Description Language
WWW	World-Wide-Web

RASSP PROGRAM
FIRST INTERIM TECHNICAL REPORT

NOTE

This report and its appendices are available electronically via World-Wide-Web (WWW) or File Transfer Protocol (FTP). Transfers are possible only from sites with domain names **.ARPA* or **.MIL*. Transfers will not be permitted from other sites, or if reverse name lookup (site cannot be verified as **.ARPA* or **.MIL*) is not functioning for a site. The Universal Resource Locators (URLs) for these Government-only restricted areas at Lockheed Sanders are:

- WWW: http://rassp.sanders.com/rassp-gov
- FTP: ftp://rassp.sanders.com/rassp-gov

Documents referenced in this report are located in the *MONTHLY_REPORTS* folder in the *JULY_1944* subfolder.

1.0 INTRODUCTION

1.1 SCOPE

The goal of the *Rapid Prototyping of Application Specific Signal Processors (RASSP) Program* is to improve the process by which complex digital systems, particularly embedded signal processors, are specified, designed, documented, manufactured and supported. The Program seeks four-fold improvements in Product Cycle Time, Life Cycle Cost, and Product Quality.

This Technical Status Report describes the accomplishments of each of the four RASSP Integrated Product and Process Development Teams during the period 1 August 1993 through 31 July 1994.

Lockheed Sanders, Inc., Hughes Aircraft Company, Motorola and ISX Corporation are currently under contract to meet these Program objectives. Personnel from these companies comprise four Integrated Product and Process Development Teams: RASSP System Engineering (RSE), RASSP Design Environment (RDE), RASSP Demonstrations, and RASSP Process Proliferation.

1.2 ORGANIZATION OF REPORT

For unity of presentation, details of achievements, lessons learned/issues and future activities are presented for each team in Paragraphs 2.0 through 5.0. Major highlights are presented in Paragraphs 1.2 and 1.3.

Paragraph 6.0 address the Technical Review Board report released at the June, 1994 review.

1.3 OVERVIEW OF ACHIEVEMENTS

The first year of the Lockheed RASSP program was completed successfully. We have smoothly transitioned our IRST demonstration from the F-22 to the F-14. More specifically:

- RASSP Methodology
  - » A VHSIC High Level description Language (VHDL)-based, top-down development approach is being put into place.
  - » VHDL style and modeling guidelines were released.
  - » The selected VHDL performance modeling tool was improved by adding additional data extraction capabilities.
  - » Performance-level modeling was incorporated into the design approach.
  - » A detailed process description was developed as the basis for the RASSP Design Environment/ Enterprise Framework (RDE/EFW) Workflow Managers.
  - » Computer-based Process Models are operational.
  - » Dataflow models and computational cost models for signal processing problems were established.
  - » Selected architectures were simulated.
  - » A set of library interface specifications were created.
- ASSP Design Environment
  - » RASSP Design Environment Release 0.1 was completed on time. This version includes a Document Manager for file check in and check out, Full Problem Reporting and Tool Launching of nearly 40 different tools.
  - » A top-level RDE architecture for Release 1.0 was defined.
  - » Tool/Technology Evaluation Methodologies were defined .
  - » Manufacturing Interface requirements were released.
  - » Build 1 was completed, along with an associated roadmap
  - » Cadence was selected as the Enterprise Framework vendor. The CAD Framework Initiative (CFI) will support this action, helping us to proceed to a CFI compliant set of interfaces.
  - » A network based communication system has been initiated. Both RASSP participants and the customer community have access to this system which facilitates communication and minimizes hardcopy documentation.
- RASSP Demonstration
  - » Changing the basic demonstration from development of a new F-22 infrared search and track subsystem (IRST) to the IRST upgrade on the F-14D is proceeding with no schedule or cost impact.
  - » A VHDL description of IRST processing and its interfaces was defined.
  - » A candidate architecture was selected and performance modeled. Models being developed include a VME bus model and an Instruction Set Architecture model of the Intel i860 processor.

- RASSP Proliferation
  - » The first release of the Visionary Demonstration ("Hot Mockup") was completed.
  - » A World Wide Web server and homepages (http://rassp.sanders.com/) were established and are accessible also through the ARPA and E/F RASSP pages. A Lockheed RASSP E-Mail Informational Address and an Information/FAX-back Line (1-800-99-RASSP) were established.
  - » A market survey was completed which strongly supports formation of a commercial company. Pay-Per-Use access to tools and the enterprise framework are featured. A prime potential product is CADEXPRESS, a Federal Express TM-like Internet-based delivery system of the Enterprise Framework and tools and tools.

1.4 OVERVIEW OF FUTURE ACTIVITIES

- The IRST flight test on the ARIMS aircraft will be completed by March 1995.
- The model development for the virtual F-14D IRST prototype will be completed in Spring 1995.
- Continued on-budget, on-schedule performance is expected for the coming year.
- Refinement and further expansion of Lockheed Sanders' RASSP process model will lead to two additional builds which in turn will drive evolution of the Enterprise Framework.
- The partnership with Cadence and associated CFI support is expected to lead to a release of the Build 1 of the EFW in September 1995. Standardization of interfaces between enterprise framework elements and the elements with which it communicates with is expected to profoundly affect the computer aided design market. This will be a major enabler of the CADEXPRESS concept.
- Benchmarks 1 and 2 are expected to be completed successfully

2.0 SYSTEM ENGINEERING ACTIVITIES

2.1 VISIONARY DEMONSTRATION

2.1.1 Goal. To produce a unified specification of the RASSP Vision which combines outputs from existing tools with interactive graphics to guide the user through a description of the RASSP Program, RASSP Process and RASSP Design Environment. This demonstration illustrates proposed functionality and user interactions with that system.

2.1.2 Development Status. Since the Lockheed Sanders RASSP team views RASSP as end-to-end support of the full development life cycle, it was agreed that the initial focus would be on:

- Project Set-Up and Bootstrapping.
- Information Management and Support

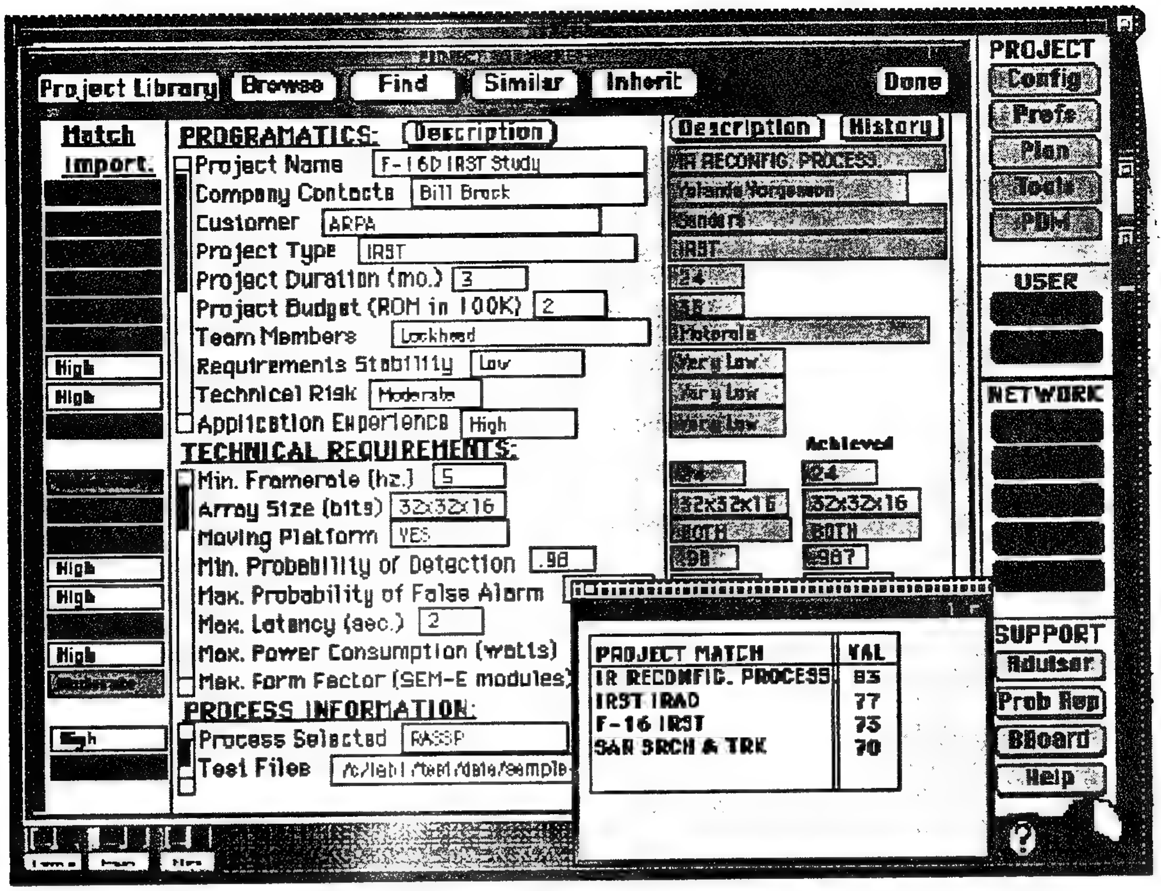
- Virtual Prototype Development

The *Project Set-Up Area* (Figure 2-1) illustrates several key themes:

- RASSP supports a wide range of users, from program managers, through systems engineers and detailed hardware/software engineers, to project data administrators and support staff.
- RASSP support extends to a geographically distributed team that includes customers (government, commercial), developers, manufacturing and packaging entities, and end users.
- RASSP's reuse and advisor capabilities are central elements in pursuing the fourfold improvements in cost, time, and quality.
- The description and requirements identification of a project using existing project databases help fill gaps in understanding the requirements and provide useful insights into the company's (teams) previous activities and experiences with similar projects.
- This segment of the visionary demonstration supports selection and tailoring of a development process model for the project, again using historical and predictive models as advisor support.
- A mapping of associated tools to each process step may be made to identify company/project standards for the development team. Upon completion, a resource manager evaluates the availability of selected tools when needed (based on a program plan interface to project scheduling tools supported by RASSP), and reports its findings. Tool shortfalls may be remedied by renting needed tool licenses via direct link from RASSP to the Electronic Information Corporation, a joint venture company formed as a result of Lockheed Sanders' RASSP which "rents" Electronic Design Automation tool licenses on a "pay-per-use" basis.

Information Management And Support focused on the rapid identification of engineering groups and review teams in concert with the process development plan selected. Key features include:

- Establishing document/data promotion and group permissions for access and manipulation of information.
- Review and modification of staffing plans and assigning key staff members to the associated working groups. The advisor provides functionality at this step by suggesting staff roles based on previous engineer experience.
- Rapid, automated establishing of group/user permissions for data and tool access, plus logins for team members.
- A problem reporting mechanism and a "bulletin-board" concept for users that provides personnel and RASSP environment notifications at the Program level, role level (e.g., hardware, software) and individual level. The bulletin-board is viewed as a key mechanism for providing users with situation awareness.



PROGRAMATICS:

Project Name	F-16D IRST Study	IR RECONFIG. PROCESS	
Company Contacts	Bill Brock	Yolanda Yorgenson	
Customer	ARPA	Sanders	
Project Type	IRST	IRST	
Project Duration (mo.)	3	24	
Project Budget (ROM in 100K)	2	38	
Team Members	Lockheed	Motorola	
Requirements Stability	Low	Very Low	High
Technical Risk	Moderate	Very Low	High
Application Experience	High	Very Low	

TECHNICAL REQUIREMENTS:

			Achieved	
Min. Framerate (hz.)	5	24	24	
Array Size (bits)	32x32x16	32x32x16	32x32x16	
Moving Platform	YES	BOTH	BOTH	
Min. Probability of Detection	.98	.98	.987	High
Max. Probability of False Alarm				High
Max. Latency (sec.)	2			
Max. Power Consumption (watts)				High
Max. Form Factor (SEM-E modules)				Moderate

PROCESS INFORMATION:

Process Selected	RASSP	High
Test Files	/c/lab1/test/data/sample	

PROJECT MATCH	VAL
IR RECONFIG. PROCESS	83
IRST IRAD	77
F-16 IRST	73
SAR SRCH & TRK	70

Figure 2-1. Visionary Demonstration Project Set-Up

VIRTUAL PROTOTYPE DEVELOPMENT (Figure 2-2) is at the core of the RASSP environment vision. This segment shows:

- An indication of the separate "views" that RASSP provides.
- How the hierarchically decomposed process view can illustrate the current process step and aggregate status.
- How concurrent engineering practices are used to provide downstream process steps with preliminary data more effectively. Appropriate downstream information is also channeled back "up stream" as needed to help direct and affect design decisions.
- How it allows status information access to supporting data and documents at each step in the process.
- How tools may be launched directly from this interface with tool encapsulation providing seamless access to data and information for a mixed set of VHDL-based tools using libraries for rapid evaluation and development.
- How access to information and tools is controlled by the Product Data Manager to ensure that data is both available and authorized for an individuals' access. The RASSP Process Manager monitors activities

including document promotion and automatically enables process steps (notifying lead engineers identified for that step) as appropriate.

- Thermometers are used to support automatic analysis and provide visual display of adherence to selected technical requirements. Thermometers can support various levels of analysis based on the level of detail available for their computation.

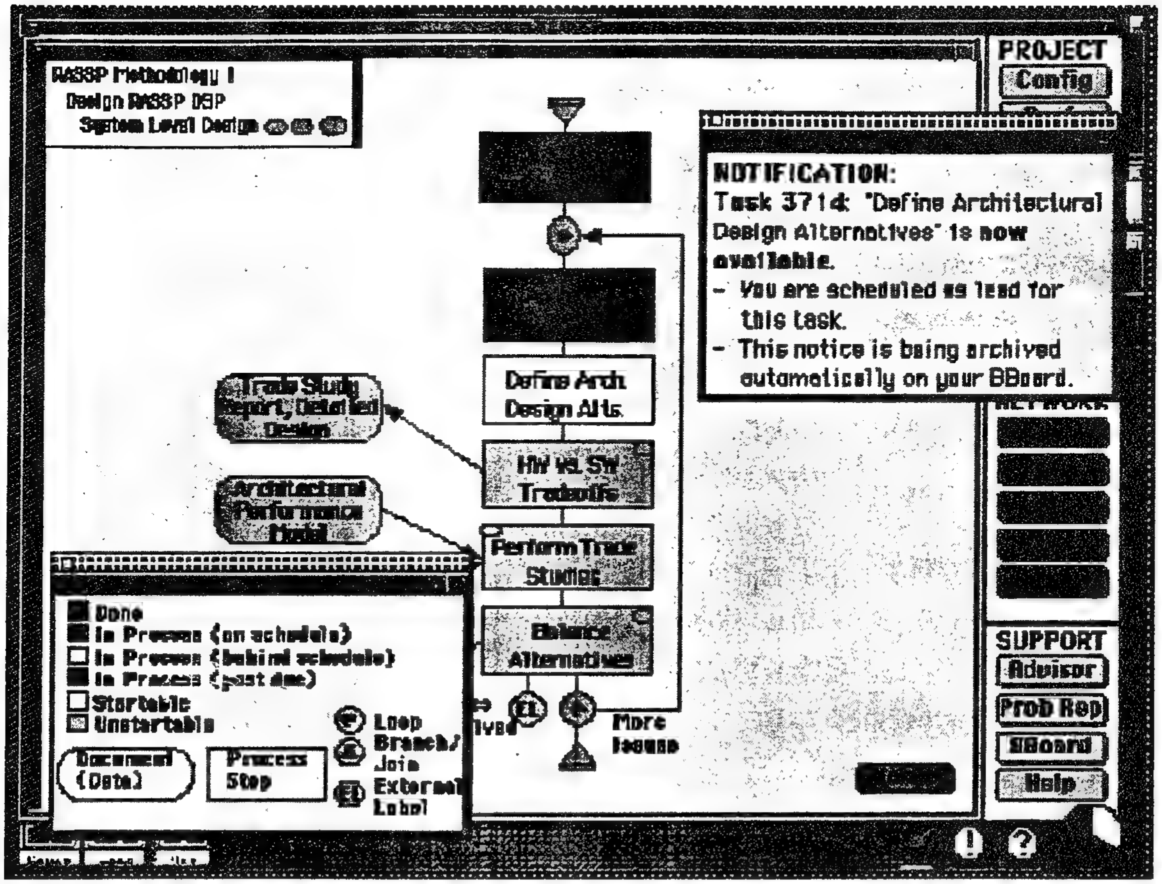

NOTIFICATION:
Task 3714: "Define Architectural Design Alternatives" is now available.
- You are scheduled as lead for this task.
- This notice is being archived automatically on your BBoard.

Figure 2-2. Visionary Demonstration Process View

2.1.3 Availability Status. The latest version *Vizdem* runs as a stand-alone application on a Macintosh computer. It requires 5 MB of RAM, a 640x480 (minimum) color screen and about 7 MB of disk space. It includes the latest version of the Program Roadmap. At this time, the Visionary Demonstration:

- Has been released periodically to RASSP team, selected government and support personnel/organizations, and key potential Beta Sites. The current development schedule supports major releases on a 6-month basis, with minor releases as needed.
- The Visionary Demonstration is available in prototype versions from the Lockheed Sanders RASSP archive machine by controlled anonymous FTP in */rassp/system_engineering/vizdemo/release/Vizdem1.9.sea.bin*.

- Public requests for a copy of the Visionary Demonstration may be made through the RASSP program office.

2.1.4 **Lessons Learned and Issues**. The Visionary Demonstration has proven to be a useful tool for describing and demonstrating Lockheed Sanders' RASSP process. Two issues must be addressed:

- Keeping it current with the vision of RASSP.
- The limited ability to capture "real" applications within the Macintosh presentation format.

2.1.5 Future Activities

- Broadening the Visionary Demonstration to provide views of the RASSP system in a larger part of the project life cycle. Emphasis will be on Model Year upgrades, maintenance and support.
- Improving the demonstration of the distributed, collaborative nature of Lockheed Sanders' RASSP environment.
- Updating and emphasizing the primary role of the RASSP process in the Lockheed Sanders team's vision of RASSP.

2.2 RASSP ROADMAP

2.2.1 Goal. To develop a RASSP Roadmap to ensure that all parties understand the interactions between program elements.

2.2.2 Status. The RASSP Roadmap (Figure 2-3) provides views of ten different aspects of Lockheed Sanders' RASSP Program:

- Programmatics
- Standards
- RASSP Functions
- Alpha-Sites
- RASSP Releases
- Beta-Sites
- Educational Tech-Base
- Demonstrations and Benchmarks
- Industrial Technical Base
- Electronic Information Corporation

For each view, up to ten types of activities may be identified, with multiple milestones for each one. Information is mouse-accessible. For example, under EDUCATIONAL TECH-BASE, selecting a specific contract brings up a document in the Microsoft Powerpoint Viewer (freeware) which includes contact information and high-level goals for each effort. The Milestones are also mouse-accessible and provide appropriate descriptive information.

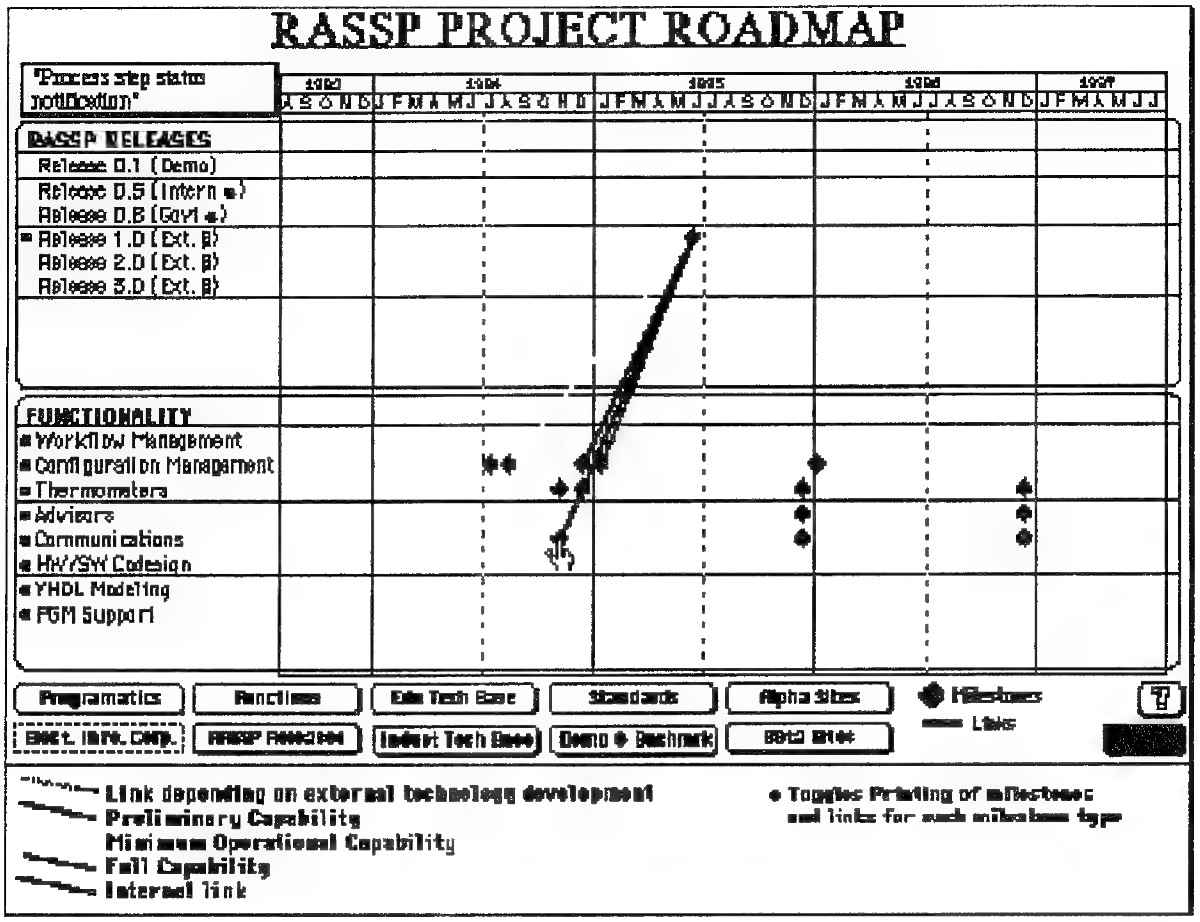


Figure 2-3. RASSP Roadmap

The most important aspect of the roadmap are the links between items that reside in different program views. For example, there is a visual link between a technology provider from the tech-base and a functional milestone that this technology supports in the *RASSP Functions* view. How and where that functional milestone applies to future RASSP Releases is also shown. Color coding of the links indicates the "maturity" of support (e.g., preliminary, minimal, complete). The links are also mouse-accessible, providing descriptions of the selected link.

Items may be turned off to declutter the displays.

Roadmap 1.0, was integrated into the Lockheed RASSP Visionary Demonstration V1.9 and demonstrated at the annual 1994 RASSP Conference. It will continue to be updated and included in the "Program View" section of the Lockheed Sanders RASSP Vizdem.

2.2.3 Lessons Learned and Issues. It is too early to judge whether the Roadmap provides a significant benefit. However, assembling the required information provided insights into the connections among RASSP program elements.

2.2.4 Future Activities

- Continue populating of Roadmap information, concentrating initially on links between Tech-Base contractors and needed RASSP functionality.
- Including of additional capabilities to support easy access to and manipulation of the Roadmap
- Providing an Internet-accessible version of the Roadmap for distributed and collaborative access and manipulation by organizations and communities involved.

2.3 ARCHITECTURE SELECTION

2.3.1 Goal. To develop and document an approach for selecting a specific architecture from a set of general building-block elements. Principal activities in this regard included:

- Completion of a *RASSP Architectural Issues* document.
- Completion of a set of architectural metrics to guide a systems engineer in evaluating alternative architectures.
- Initial specification of a *RASSP Architecture Guide* which summarizes the architecture selection process.

2.3.2 Status

2.3.2.1 ***RASSP Architectural Issues Document***. This document was published in July, 1994. From January through June, it underwent monthly updates. The extensive background material in this document served as the starting point for developing the RASSP architecture selection process.

A paper based on material in *RASSP Architectural Issues* was submitted to the conference on *Advanced Signal Processing: Algorithms, Architectures, and Implementations V* during the International Society for Optical Engineering's 1994 International Symposium on Optics, Imaging, and Instrumentation (24-29 July 1994, San Diego, California).

2.3.2.2 ***RASSP Architecture Metrics Document***. This document, published 20 June 1994, proposes a set of 18 metrics to evaluate the quality of a signal processor architecture and implementation. These metrics were developed by collecting signal processor requirement specifications, extracting key parameters that affect architecture selection, generalizing the parameters into a set of RASSP metrics, and developing an approach for applying relative weighting factors. Consequently, these metrics can be used to flag missing requirements data, compare alternate design implementations to derive an acceptable solution, or select a preferred solution among several choices.

This document also contains rationale for the metrics and weighting methodology, a discussion of the weighting methodology, an example showing how specific signal processor requirements are converted into a set of weighted metrics, and a visual presentation of metric values as a set of "thermometers" displayable on a RASSP Design Environment workstation.

This is a visionary document, proposing how the set of metrics might be used by a RASSP designer. The intent is that a designer be able to map a tentative architecture and get back where it meets the requirements, where it fails, and by how much. Issues still to be resolved include how to handle incomplete design information and how to measure uncertainty in the computation of the metric values.

2.3.2.3 ***RASSP Architecture Guide*** **Document**. This document will provide RASSP Process users with documentation to facilitate understanding the RASSP approach to initial architecture selection. The first draft, scheduled for release in November 1994, includes rationale for the set of RASSP architecture candidates and the architecture selection process, a list of architecture candidates, a description of the architecture selection process, and an illustrative use of the process.

The Guide was prepared by reviewing established signal processor architectures, generalizing them into a set of RASSP candidate architectures, and postulating generalized methods of selecting a preferred architecture based on system requirements and implementation technology. Work to be done includes finalizing the illustrative architecture selection example.

2.3.3 **Lessons Learned and Issues**. One important issue was identified: the need for an open system interface between the attached processor and the network interface unit in a scaleable parallel processor. This standardized hardware/software interface must clearly separate the attached processor from network communications, allow any vendor processor to plug into a network, and facilitate network updating with minimal impact on existing processors. This interface has not yet been standardized in the commercial industry, although multiple company-proprietary implementations exist. While RASSP will have the ability to adapt to any known well-defined interface, the use of preferred interface standards will be encouraged.

2.3.4 Future Activities

- Define an open system interface for processors.
- Develop test strategies for architecture selection.
- Develop a set of system test strategies that can be applied to the various architectures as required by the system/user requirements.
- Support open system interface development by participating in the Scaleable Coherent Interface (SCI) (IEEE 1596-1992) extension meetings. We plan to uncover and resolve issues related to the interconnect network by focusing on SCI as an example of an important class of emerging bus

standards SCI has been proposed as a real-time heterogeneous parallel processor interconnect network by both the Navy's Next Generation Computing Resources (NGCR) Program and the Navy/Marine/AF JAST Program. The SCI Real Time Extension Working Group (P1596.6) is considering modifications to the base SCI specification to support military needs for priority and fault tolerance.

- Coordinate open system interface development with other military development programs to maximize effectiveness and provide an opportunity to spread the RASSP methodology. Specific programs of interest include CHAMP, ATR and JAST. The need for an open system interface has already been presented to NAVAIR by Lockheed Sanders.

2.4 ARCHITECTURE EXPERIMENT

2.4.1 Goal. To demonstrate that signal processing systems can be designed using an architecture comprising fundamental signal processing modules with a common interface. These modules, implementable in hardware or software, realize a system directly from its design at the data flow level. This approach permits independent implementation of each module; i.e., individual modular implementations can be interchanged without adversely impacting overall system design. The implementation will support all Processing Graph Method (PGM) semantics. An important goal of the experiment is to learn the practical aspects of defining and using modules with common interface. This interface is intended to be open and non-proprietary so that the software can be easily ported and reused.

2.4.2 Status. To date, a signal processing package was developed which implements this common interface. The goal was a software representation that hides implementation choices and inter-processor communications from the high level functional description (data flow graph). The software architecture is object-oriented, implemented as a set of classes coded in C++. Figure 2-4 shows the hierarchies of classes comprising the software architecture. The shaded classes are not yet been fully implemented, but are essential to the expansion of the architecture to handle a wide variety of applications. The design incorporates near-term requirements and provides for future flexibility and expandability.

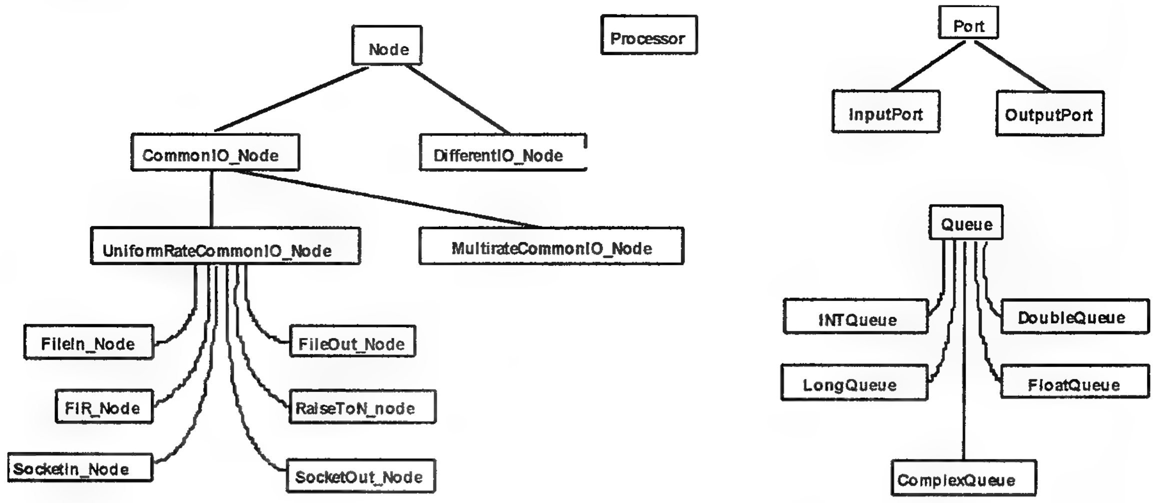


Figure 2-4. Software Architecture Classes

The first successful demonstration (Figure 2-5) consisted of a simple uniprocessor application. The application comprised four Nodes in series, all executing on the same processor:

- *FileInNode* (reads data from a file)
- *FIR_Node* (performs FIR filtering)
- *RaiseToN_Node* (raises input to the Nth power)
- *FileOutNode* (writes data to a file).

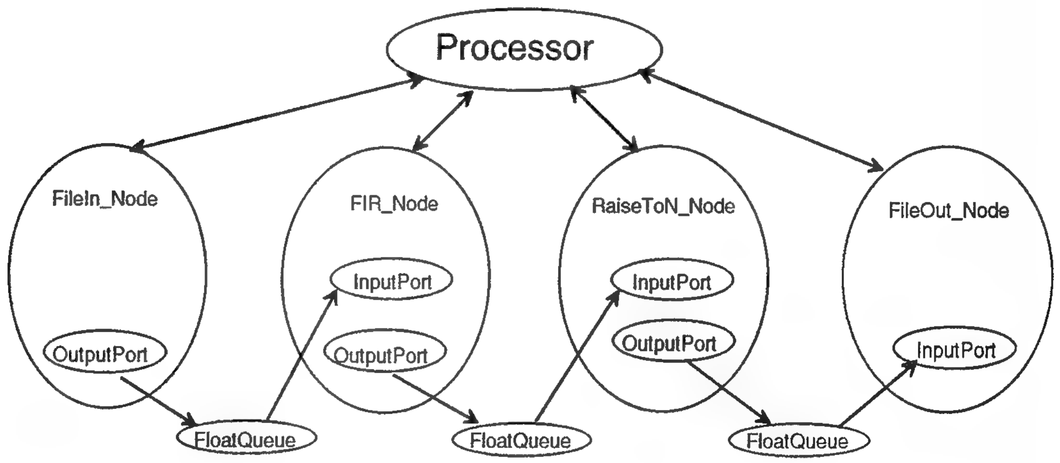


Figure 2-5. Typical Dataflow Graph Showing Node, Port and Queue Objects

Comparison of output data from the application with results calculated by Matlab revealed minor round-off differences. In this application, some implementation details were simplified; e.g., selecting Queue sizes, scheduling algorithms and minimum/optimal/maximum values for Node firings. The software will be modified in the upcoming month to allow for "better" or more flexible choices in the implementation for these simplifications.

The second demonstration (Figure 2-6) consisted of five nodes in series, distributed across two processes executing on separate processors.

The Naval Research Laboratory's Processing Graph Method (PGM) and the University of California (Berkeley) Ptolemy tools played a significant role in developing the software architecture. PGM's semantics used to model the dataflow, served as a baseline for the software architecture. Ptolemy's object-oriented approach in C++ was also influential.

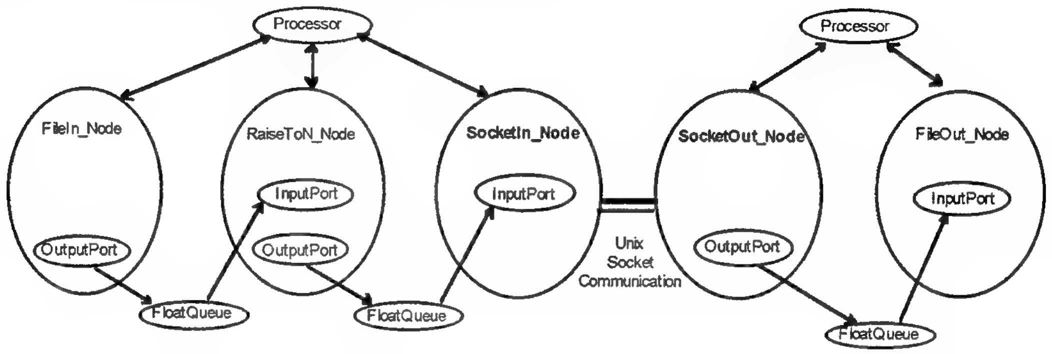


Figure 2-6. Typical Dataflow Graph Showing Multiprocessor Mapping

2.4.3 <u>Lessons Learned and Issues</u>. Several issues must be addressed:

- <u>*Hardware/Software Duality*</u> - This concept of must be tested by porting the software architecture to different platforms, in which some functionality is implemented in hardware. It also must be tested by porting the code to Ada preprocessors (e.g., Cfront). This endeavor will also uncover issues related to operating system support and interprocessor communications (via means other than sockets) on different platforms. No design decisions have been made which would preclude a port to Ada 9X; this port is part of the future plans.
- <u>*Code Size and Run-Time Efficiency*</u> - This area requires investigation because signal processing applications must execute in real-time and/or in a limited memory space. By porting to different processors and profiling performance, an accurate assessment can be made.
- <u>*Implementation Precision*</u> - Until now, all of our implementations have assumed 32-bit floating point arithmetic. Future studies must address how to deal with varying bits of precision and fixed *versus* floating point computations, as these are prevalent in DoD systems.
- <u>*Integration of Software with A Graphical User Interface*</u> - In future demonstrations, the graphical user interface must not only display the data as processed, but also provide feedback to the user on the effectiveness and efficiency of the implementation. An interface to Ptolemy is being seriously considered to provide a means of running simulations and generating code for various processor configurations for DSP algorithms.

2.4.4 Future Activities. The immediate goal is to develop a multiprocessor implementation that will be initiated as a single process, partition the graph as specified, launch processes on remote machines to execute the various portions of the graph, and oversee the establishment of the communication between the processes. Ideally, the user need only specify the application nodes that are needed; *SOCKETOUTNODE* and *SOCKETINNODE* will be established automatically where needed. A port of the software to Ada 9X is also being planned, as the standard emerges.

2.5 METHODOLOGY DEFINITION AND METRICS

2.5.1 Goal. To define a process for defining, designing and supporting digital signal processor units. This process, as it evolves during the RASSP Program, will lead to four-fold improvements in cost, schedule and quality. To evaluate its effectiveness, metrics will be devised and used to measure performance. New releases of the RASSP Process will be incorporated in successive releases of the RASSP Design Environment, and the collection of metrics will be enhanced through automation.

2.5.2 Status

2.5.2.1 ***RASSP Process Development***. The RASSP System Engineering Team collected process information from each team member company, and developed an initial RASSP System Development Methodology. A top-down VHDL development flow (see Paragraph 2.7) and the RASSP concept of a Virtual Prototype are being incorporated into the process

Three top level development methodologies (spiral, waterfall and incremental) were then captured in a RDD 100 requirements capture and modeling. These models were simulated to verify accuracy, and the output incorporated into the RASSP Visionary Demonstration computer program. These models were then developed into greater detail, using best practice inputs from all three RASSP Team member companies, and incorporated into a single overall process flow. The result of this effort is our initial RASSP Process

The RASSP Process documents design development from initial customer requirements, through system design, algorithm refinement, hardware and software co-design activities, integrated test development, and integration and test. A "boxology" format is used for both Methodology and Detailed Task levels; i.e., individual tasks are represented as single boxes along with input and outputs definitions, exit criteria, applicable tools, task description and hierarchical reference.

Some areas (e.g., VHDL modeling and hardware development) were included in great detail, with much emphasis on a VHDL Top-Down design flow definition aimed at meeting the RASSP SOW requirements, and creating a useful Virtual Prototype. Some hardware areas were defined as *process modules*, to be used only as applicable; e.g., the MCM process module is used only when MCM is selected at hardware partitioning level). In addition, an RDE tool environment was created, showing the flow of CAD data files among the various tools listed in the Methodology and Detailed Task levels.

Future work will involve adding detailed descriptions of the environment and the database, and metrics detail to the Methodology and Detailed Task levels.

2.5.2.2 ***Rapid And Disciplined Development***. The RSE Team developed a modeling technique that replicates work actually in an interactive concurrent engineering environment. Work functions are presented as the effort needed to transform an input package into an output package; e.g. Requirements Analysis, which transforms a requirements document (input) into an analysis document (output). Defining the receipt of input packages as asynchronous events, and allowing incomplete packages to passed from activity to activity results in a close modeling of the work environment of a co-located design team, which shares preliminary data to enable "downstream" activities to make earlier progress.

This technique was presented at the six month review. Paragraph 3.4 contains more details on process simulation.

Note that this RASSP technique differs sharply from the traditional process modeling, which either contains a "waterfall" methodology (every activity must be completed before the next activity can start) or a "built-in iteration" methodology which predicts the number of process iterations, in effect another version of the waterfall approach.

2.5.2.3 ***Metrics***. The RSE and RDE Teams developed a set of metrics for the RDE to automatically collect in RDE Release 1.0. An electronic copy will be placed on the RASSP server and will describe the subset implemented for RDE Release 0.1, the metric units, and how they will be collected in RDE releases. Product Metrics are presented at a high order only; more metrics will be added, and incorporated in the RDE database that supports the System Evaluation and Estimation of Resources (SEER) cost model in RDE release 1.0. Separate metrics are presented in the *Architecture Metrics Document* (Paragraph 2.3.2.2) which describes how system level metrics can be used for architecture selection, and how they can be used with weighted input requirements to create an effective evaluation of different architectures.

2.5.3 **Lessons Learned and Issues**. The principal lessons learned were:

- It is difficult to find a process representation that satisfies all user needs. Requirements for RDE, management overview, designer guidance and database population are vastly different. Combining different views of one process to meet these varied needs at differing levels of abstraction is a great challenge.
- Even though the "-illities" are included in the Design Guideline documents and referenced by appropriate process activities, they need more emphasis and more detailed contents.

- The transition from a general purpose process to a lower level process with the product content identified (e.g., six modules, two containing MCMs, one MCM containing custom ASIC) is difficult to represent.
- The process needs for a work flow manager are not well known at this time, and state-of-the-art work flow management is not yet well developed for practical cases.
- Process definition must be flexible enough to allow engineers to do a better job across a wide spectrum of jobs without telling them how to do design at every step.

Six principal issues must be addressed:

- A consistent representation of the required contents for a RASSP Process must be developed with all customer requirements incorporated at the appropriate level of detail.
- Selection of a work flow management tool.
- How to conduct effective reviews in a concurrent engineering environment without slowing down the design effort.
- Relationships with existing MIL specifications, industry standards, and best commercial practices.
- Improvements in metrics which predict or measure product quality and in metrics which predict or measure product quality and in metrics which predict or measure process performance.
- How to emphasize the RASSP "valued-added" intellectual content within the process.

2.5.4 Future Activities

- Defining a RASSP Process representation that meets all customers needs.
- Integrating and automating instantiations of the RASSP product structure and lower level detailed processes and product metrics..
- Developing a RASSP Metrics List and associate the list with the RASSP process for automation.
- Publishing the *RASSP Design Guideline documents*
- Evaluating new tools and process modifications that will support RASSP goals.

2.6 PROCESS MODELING

2.6.1 Goal. Process simulation characterizes individual tasks in terms of time to complete the activity. It is assumed that the activity models and, hence, the process itself, are random and unpredictable. This, coupled with the anticipated process complexity, precludes any analytically tractable solution of process cycle time. Process simulation permits two types of analysis to be performed. First, the process can be analyzed for completeness, that is, the process should result in producing the intended output. Second, cycle time analysis can be performed by modifying the process and running the simulation to determine the effect on cycle time. The first objective relates to checking that a process description is valid, the second is targeted at optimizing a valid process description relative to cycle time.

2.6.2 <u>Status</u>. The first task was to construct an abstract model of the processes. Since the simulation plan was concerned primarily with time to complete the process, the rate of work necessary to complete a task was parameterized. The parametric value reflected the history of that particular task; i.e., work rate for original work differs from re-work. The model was set up to consider feedback in the process and to measure the effect of early *versus* late error/problem discovery in a process.

The model was implemented using i-Logix's *Statemate*. This tool was selected for two reasons. First, a process comprising a series of tasks related by data that passes from one task to the next appeared easily modeled by a state process representation. Second, Statemate allows the customization of user display panels; this would permit constructing a meaningful method to relate the information to the investigator. A proof of concept for this implementation was presented at the biannual review in February, 1994.

Development of the proof of concept quickly revealed shortcomings in the choice of implementation, and a search began to find a tool that specifically simulated processes. Based on a list of criteria, two commercial tools were evaluated:

- *<u>RDD-100 (Ascent Logic)</u>* - Originally designed as a requirements tool, it initially seemed to be among the better ones available for this application despite its complexity which requires a fair amount of learning time. RDD-100 satisfactorily supports the decomposition of requirements. However, its process simulation capability appears to have been added as an afterthought. For example, various non-intuitive rules that must be followed when defining a process destined for simulation makes building a complicated process very difficult. The debugging capability, although useful, requires much improvement. Also, there is no clear, direct way to parameterize individual tasks. Finally, RDD-100 contains no explicit utility for Monte Carlo simulation.
- *<u>CACE-PM (Perceptronics)</u>* - Originally designed to simulate processes, it is easy to use and has a very intuitive user interface. Definition of processes for simulation is straightforward. Its major drawback relates to the structure and implementation of the tool itself. For example, variables for individual tasks are global which, in addition to being bad programming practice, causes problems for large process definitions. Also, the user cannot filter simulation data or post analysis. This causes lengthy execution times and massive data writing to disk for Monte Carlo analysis.

2.6.3 Lessons Learned and Issues. The principal lessons learned were:

- Although similarity exists between tasks and processes as compared to states and state processes, a state process simulation capability does not necessarily provide acceptable capability for process simulation.
- Process simulation tools are still evolutionary in terms of capabilities. They provide basic functions, such as process consistency checks, but do not necessarily support more complicated actions, such as process reconfiguration, probably because process engineering is an active area of research. As process engineering matures, so too will process simulation tools.
- Process simulation tools are essentially stand-alone products. RDD-100 and CACE-PM will not interface with other tools that deal with processes (Perceptronics *is* developing process management capability for its tool).
- The level of process description is critical. Processes can neither be constructed nor improved by ignoring lower level structures. Conversely, process descriptions at low levels require some visibility into product structure and development strategy.
- Process engineering for developing custom systems, which reflect schedule and development, is much different than process engineering for developing technologies, such as ASIC, which emphasize methodology.

Four principal issues must be addressed:

- Inclusion of a process simulation tool in the RDE toolset has complicated the selection of the tool.
- Importing and exporting process descriptions with RDD-100 and CACE-PM is not possible. This is important if the tool will be coordinated with other tools; e.g., the RDE where process descriptions will be used by a process manager tool.
- Validation of process simulation models will be difficult. The difficulty goes beyond collecting metrics from previous process simulations and parameterized models based on the metrics.
- The overall usefulness of process simulation is not well established. At a minimum, the tool should validate a process. Whether, the process can be optimized is debatable. This exercise has the potential for a huge return on investment. It may also result in a dead end.

2.7 VHDL MODELING

2.7.1 **Goals**. The primary goals of the VHDL Modeling task are to:

- Define the basic VHDL modeling methodologies to be used in RASSP work.
- Establish a top-down VHDL modeling methodology that will simulate system behavior using only a VHDL simulator and an Ada compiler.
- Develop an in-depth understanding of these methodologies and the tools to support them.

VHDL modeling offers a standard language and a full set of simulation capabilities. It is central to the RASSP development philosophy. It will provide:

- A system description to be used in verifying system behavior prior to system fabrication.
- A storable definition of system behavior and interfaces.
- An specification for system interfaces and functions such that model year upgrades can be developed from the specifications.

2.7.2 Status

2.7.2.1 ***VHDL Performance Modeling***. Candidate architectures are modeled with a set of basic items such as: processors, memories, interconnects, and I/O devices. Software is modeled as a set of tasks, each of which the user analyzes to characterizes in terms of processor resource and memory utilization. Processors are characterized in terms of execution and latency times for basic operations such as "FFT" or "Handle message at a high level or "Add A&B" at lower levels. Objects in the architecture interact by exchanging tokens (usually without data). The architectures are then analyzed in terms of utilization, throughput and latency.

Honeywell VHDL performance models were evaluated. These models, developed for the GDP project under Air Force sponsorship, were given to the Demonstration Team for their use. A comparative performance modeling analysis was performed using OpNet, a commercial, non-VHDL tool. OpNet provides models similar to VHDL performance models, and provides similar architecture measurements. OpNet does not, however have a built-in software model or token passing model. These must be developed by the user.

Use of OpNet and similar tools, speed up the design process because of their graphical user interfaces and high speed "validated" robust libraries. However, though their use results in a tool specific system description. Process tailoring guidelines may suggest using commercial tools where development cost is the major issue and the VHDL models where vendor Neutral documentation is most important.

2.7.2.2 ***ISA Model Development***. The System Engineering and Demonstration Teams cooperatively defined the ISA modeling approach and oversaw the Georgia Institute of Technology's development of an ISA model for the i860. This model:

- Contains 4,000 lines of VHDL code.
- Runs in a Vantage simulator on a Sparc10, executing 1,300 instructions per second in its standalone mode, and 300 instructions per second when wrapped with a bus interface model to create a full bus functional model.

The model does not include i860 graphics instructions or trap handling mechanisms. Graphics instructions are not needed for RASSP's current plans and are not scheduled for development. Trap handling is being developed.

ISA modeling methods are described in detail in *Rapid Prototyping of Digital Systems with COTS/ASIC Components*, Famorzadeh, S. *et al.*, PROCEEDINGS OF THE FIRST ANNUAL RASSP CONFERENCE.

2.7.2.3 ***VHDL Simulation Tool Evaluation***. A set of VHDL simulation enhancements were evaluated. The goal was to quantify performance improvements available from multithreaded VHDL simulators and by using very high performance computers for VHDL modeling. Three test cases were developed, comprising 3,000, 60,000 and 150,000 lines of VHDL code each. Each test case used mixed levels of abstraction.

Table 2-1 lists processing results to date. These results show that, as expected, multithreaded simulations do not yield large performance gains for the small test case since it contains a limited number of VHDL processes. Multithreaded simulations provide more substantial performance gains for the medium and large test cases. The CRAY 6400 Superserver provided about 5:1 improvement over the SPARC 10 for the medium case. These results need to be completed and the memory usage, computer costs, and disk space requirements for these simulations need to be collected.

TABLE 2-1
CURRENT VHDL SIMULATION TEST CASE RESULTS

SIMULATOR TECHNIQUE1 (HW/SIM TECHNOLOGY)	VHDL TEST CASE SIMULATION TIME (~SEC)		
	SMALL[2]	MEDIUM[2]	LARGE[2]
SPARC 10	20.16	4279.6	72,000
SPARC 10/1 Thread	14.45	4150.5	70,000
SPARC 10/2 Thread	10.02	2510.6	32,000
SPARC 10/ 3 Thread	8.53	1230.0	21,500
SPARC 10/4 Thread	8.42	1050.0	17,500
CRAY 6400/1 Thread	In Process	850.0	In Process
CRAY 6400/2 Thread	In Process	In Process	In Process
CRAY 6400/3 Thread	In Process	In Process	In Process
CRAY 6400/4 Thread	In Process	In Process	In Process
CRAY 6400/12 Thread	Not Planned Yet	Not Planned Yet	Not Planned Yet
VIP Paradigm	Not Planned Yet	Not Planned Yet	Not Planned Yet

1. SPARC 10 hardware uses Vantage Spreadsheet 4.1.3; all other threaded simulations use Vantage SpeedWave/MT.
2. Small, medium and large test cases comprise 3,000, 60,000 and 150,000 lines of code, respectively.

2.7.2.4 ***VHDL Process Definition***. A top-down VHDL modeling approach was developed that includes performance analysis, full functional simulation, and detailed design. Figure 2-7 shows a top-level view of this process, details of which are represented in *Processes and Experiences in VHDL Top Down Design*", Dreiling, R., PROCEEDINGS OF THE FIRST ANNUAL RASSP CONFERENCE.

2.7.3 Lessons Learned/Issues

2.7.3.1 VHDL Performance Modeling. The VHDL performance models:

- Did not yield the promised analyses.
- Require heavy computer resources, so much so that a 32 node system cannot be simulated on a Sparc 10 with 128 MB of RAM.
- Validated the architecture selected for the demonstration.
- Required significant effort to characterize the software for the performance models. The Honeywell models' software description is very fine grain; i.e., describing software almost at the instruction level rather than the function/delay level.

The OpNet/ VHDL performance modeling approach comparison indicated that:

- The VHDL tool has a predefined approach to modeling software. This is a benefit when the approach matches the problem. OpNet does not a have predefined approach. This makes it more flexible, but requires developing such an approach from scratch.
- OpNet has several built-in models, but they are not well suited to signal processor development because they focus on large-scale communications systems.
- The VHDL tool supplies only the most generic of models. The RASSP Engineering Database seeks to specific models.
- OpNet has more built-in analysis tools than the VHDL tools and these tools are aimed at performance characterization.
- The VHDL tool uses standard VHDL simulation wave traces and print statements to collect statistics. These do not necessarily match the information that the system architect wants to obtain from the performance simulation. Thus, RASSP has added post-processing tools to the VHDL simulation output.
- OpNet has more tools to automate the analysis process by scripting runs across parameter values.
- OpNet uses a proprietary data representation unlike the VHDL performance modeling tool.

Overall, OpNet tool provides a useful way of doing performance analysis, but its proprietary data representation generally limits its use on the program.

2.7.3.2 ***ISA Model Development***. The key lesson learned in this area is the necessity to put software debugging utilities into the ISA microprocessor model; e.g., those typically found in a debugger or an in-circuit emulator such as, dump registers or memory or set breakpoints. These elements are usually not included in currently available VHDL hardware component models and, as a result, the simulation results do provide software engineers with any usable information. About half of the i860 ISA model development effort comprised adding software debugging elements.

2.7.3.3 ***VHDL Simulation Tool Evaluation***. For the selected test cases, multithreaded simulations provided significant improvement over the medium and large test cases. However, performance improvements in both the multithreaded simulators and the Superserver were less than an order of magnitude. Significantly greater improvements are needed if large (hundreds of processors) multiprocessor simulations are going to be successfully performed for large amounts of data. The utility of such simulations is still under assessment and next year's process definition work will include studying alternative approaches. In particular mixed level simulating where much of the Target system is modeled at a higher level of abstraction (thus requiring less computation) and only a small portion of the target is modeled in detail holds considerable promise.

2.7.3.4 ***VHDL Process Definition***. Multiple views must be developed for a process. The process definition work defined an abstract process which cannot be used directly in a workflow manager. Before they can be used effectively in the RDE, process definitions must be developed that encompass tools and are tailored to the product being developed .

2.7.4 Future Activities

- Analyses of VHDL modeling tools will be completed. This will include completing an assessment of the Cray Superserver, and examining Zycad's ViP VHDL accelerator and Vantage's Optium simulator.
- Alternative VHDL simulation approaches will be evaluated, such as Redwood's cycle-based simulator and the University of Cincinnati's QUESTsim parallel discrete event simulator.
- Alternative hardware/software codesign methods will be evaluated that do not require using VHDL ISA models yet still result in complete and non-proprietary models.

2.8 SOFTWARE INNOVATIONS

2.8.1 **Goal**. To collect existing information on specific software topics that could be used in improving the RASSP software methodology and the RASSP design process. The task is not to update the RASSP design process or the RASSP software methodology, but it is to organize information that could be used in the future updates.

2.8.2 **Status**. Of several areas initially identified for possible investigation, two areas were chosen for exploration:

- The nature of digital signal processing software.
- Software Reuse

The first study addressed the differences between digital signal processing (DSP) software and other types of software. Information sources included both DSP software designers and current literature. The study revealed that:

- DSP software stresses available resources (memory, processing power, input/output).
- Software for actual signal processing may be heavily burdened in satisfying extraneous functions (control, status, error conditions).
- Historically, DSP software engineers must be intimately familiar with the platform that will eventually host the software.
- Software designers and algorithm designers often have different objectives.

The study also identified several trends in DSP software development which, in terms the study conclusions, will result in technology improvements (e.g., tools, libraries) that will enhance DSP software the development. This initial evaluation will be followed up by the selection of a set of innovations to be applied to the software development process

The Software Reuse study revealed that extensive work has been done in this area, including PGM. A number of reuse methodologies exist with documented improvements in quality, cost and cycle time. A RASSP reuse methodology could be developed using this previous work. It is apparent that a successful reuse methodology must be applied throughout the design effort, including hardware, system and algorithm development.

Further investigation uncovered an ARPA program called *Software Technology for Adaptable, Reliable Systems (STARS)* which focuses on software reuse. STARS has developed a reuse library called *A Source for Software Engineering Technology* (ASSET) that is available to any software developer. This library contains reuse methodologies, standards, guidelines and blueprints for developing reusable software.

2.8.3 **Lessons learned and Issues**. Two principal lessons were learned:

- A reuse methodology must be developed which follows the RASSP design process. This is essential for RASSP to achieve its targeted improvements in quality, cost and cycle time.
- Reuse Library guidelines and specifications must be developed.

2.8.4 **Future Activities**. The reuse library will be developed and associated with the RASSP Engineering Database. Tasks include developing a library structure , specifications and a reuse methodology. STARS will serve as guide.

2.9 RELEVANT PUBLICATIONS

- *RASSP Architectural Issues*, Revision H, 1 July 1994, Document AVY-L-S-00080-101-H. Available in PostScript format on the RASSP SERVER AREA: RASSP/SYSTEM_ENGINEERING/ARCHITECTURE ARCH_ISSUES_H.PS (text) and ARCH_ISSUES_H_COVER.PS (cover page).

- *RASSP Architecture Metrics*, Revision A, 20 June 1994, Document AVY-L-S-00076-101-A. Available in PostScript format on the RASSP SERVER AREA: RASSP/SYSTEM_ENGINEERING/ARCHITECTURE ARCH_METRICS_A.PS (text) and ARCH_METRICS_A_COVER.PS (cover page).
- *SPIE Paper*, July 1994. Included in the SPIE 1994 PROCEEDINGS. Available in PostScript format on the RASSP SERVER AREA: RASSP/SYSTEM_ENGINEERING/ARCHITECTURE SPIE_PAPER.PS.
- *RASSP Architecture Experiment Plan*, 27 May 1994, Document AVY-L-S-00067-101-A. Available on RASSP SERVER: /RASSP/SYSTEM_ENGINEERING/ARCHITECTURE/EXPERIMENT/PROPOSAL.FM.
- *Tool Evaluation for RASSP Architecture Experiment*, 31 May 1994, Document AVY-L-S-00068-101-A. Available on RASSP SERVER: SYSTEM_ENGINEERING/ARCHITECTURE/EXPERIMENT/TOOLS_EVAL.FM.
- *Evaluation Comments Concerning PGM and the PGSE Environmen*t, 20 June 1994. Available on RASSP SERVER: SYSTEM_ENGINEERING/ARCHITECTURE/EXPERIMENT/PGSECOM.FM).
- *STARS World Wide Web Homepage URL.* Available on HTTP://WWW.STARS.BALLSTON.PARAMAX.COM.
- *ASSET World Wide Web Homepage URL*. Available on HTTP://SOURCE.ASSET.COM.
- *World Wide Web Virtual Library Software Engineering URL*. Available on HTTP://RBSE.JSC.NASA.GOV/VIRT-LIB/SOFT-ENG.HTLM.
- *Description of RDE Release 1.0 RASSP Engineering Database(REDB),* Revision A.

3.0 RASSP DESIGN ENVIRONMENT ACTIVITIES

3.1 RDE 1.0 ARCHITECTURE AND BUILD 1

3.1.1 Goal. To produce a design environment that is driven by user requirements, current tools and current technologies; and is characterized by a flexible methodology as defined and periodically updated by RASSP System Engineering.

3.1.2 Status. Extensive work was accomplished on the RDE System Architecture 1.0 area by forming a special Requirements and Architecture Team comprising RASSP System Engineering and RDE personnel. Weekly meetings focused on planning, decision making, and coordination of activities. To resolve especially difficult issues, 2-day weekly meetings were held over a 1-month period alternating between Scottsdale, Arizona and El Segundo, California. Weekly teleconversations dealt with specific issues. The Team utilizes an Issue-Based Information System (IBIS) which documents issues, comments, questions and answers or positional statements. IBIS is regularly updated and placed on the server.

The RDE 1.0 Build 1 operational scenario (Cadence framework services combined with RASSP developed RDE functionality) were presented at the August, 1994 RASSP Conference. Release 0.1 has been completed, installed, validated, and released to the Demonstration Team. This environment can support virtually any tool suite that runs on a SUN operating system, and is not limited to a fixed set of tools. This release includes seven major services:

- Common User Interface
- Configuration data management
- Translations; e.g. EDIF 2.0
- Tool Encapsulation
- Framemaker hypertext process flow documentation
- Metrics collection
- Public Domain Database

Release 0.1 includes Matlab and Cadis and a tool encapsulation environment to launch tools, and two schematic capture tools. The metrics that can be automatically collected at this time include tool utilization (both clock time and keystroke monitoring), tool idle time and CPU utilization. Release 0.1 has an initial database structure that allows selection of one of three databases implemented in RDE:

- *Exodus* - A public domain database
- *Sherpa* - A relational database
- UNIX File System

The UNIX system was implemented primarily for test purposes. It will not have the functionality of the commercial databases. Sherpa is still established among the teams with Hughes as server and Lockheed Sanders and Motorola as clients.

The RDE Roadmaps for 1.0, 2.0, and 3.0 were generated, each comprising a list of capabilities and features allocated to the respective model year.

3.1.3 Lessons Learned and Issues

- Team building requires time. Cultural differences, a lack of common definitions, different problem solving approaches, and inclinations toward certain tools slowed the consensus process. The RDE Requirements and Architecture Team, formed out of necessity to define and document RDE functional requirements and computer system architecture, has contributed significantly to this process.

- Sherpa is a Product Data Manager, but beyond that it does not yet support the required user interface, intertool communication, and process management services.

3.1.4 **Future Activities**. RDE Release 1.0 is scheduled for release in May, 1995. This version will support all Release 0.1 capabilities, plus work flow management, electronic inspections and RASSP Engineering Database functions.

3.2 ENTERPRISE FRAMEWORK EVALUATION

3.2.1 **Goal**. To select an Enterprise Framework that satisfies the RASSP Requirements Document in the critical areas of common user interface, document management, tool integration and encapsulation, communication management, and work flow management.

3.2.2 **Status**. A set of Enterprise Framework requirements was codified in a Request for Information (RFI) and sent to eight prospective framework vendors. From the responses, four candidate vendors (DEC, Intergraph, Altium, and Cadence) were selected and evaluated in-depth by a team comprising Lockheed Sanders, Hughes and Motorola RASSP personnel

Based on the evaluations, an Enterprise Framework from Cadence was selected and a demonstration was developed and instantiated on the Cadence Work Flow Manager for the August RASSP Conference.

3.2.3 **Lessons Learned and Issues**. Studying enterprise frameworks led to an understanding and documentation by RASSP personnel of the components and capabilities required for RASSP. There is no available commercial product that meets more than 50% of those requirements. This in turn led to the partnership with Cadence.

3.2.4 **Future Activities**

- Develop a Statement of Work for a partnership with Cadence Enterprise Frameworks and a Mentor relationship in return for tools.
- Continue to meet with Cadence to review upcoming product releases.

3.3 TOOL EVALUATION AND SELECTION

3.3.1 **Goal**. To develop tool evaluation criteria and to apply these criteria to the selection of tools that will contribute to design environment productivity.

3.3.2 **Status**. Tool evaluation methodology / criteria were developed. Based on this, a spreadsheet of 40 tools (IDE plus some requests from the System Engineering and Demonstration teams) with associated priorities (high, medium, or low) was generated and submitted for team evaluation and integration. The priority was the urgency of obtaining tools based on the planned integration schedule with inputs from the demonstration team. This spreadsheet included need dates, teams requiring (Systems Engineering, RDE, Demonstration), sites requiring (Lockheed Sanders, Hughes, Motorola), acquisition status and installation status.

A RASSP Baseline Design Methodology was then developed and over 60 tools were mapped to. Documentation was completed on schedule for 35 IDE tools (Table 3-1) in five major areas:

- Project Management
- Integration
- System Engineering
- Software Engineering
- Hardware Engineering

Related activities included:

- Generating tool cost estimates for the life of the program.
- Obtaining signed agreements from Vantage and IKOS for free use of their tools.
- Reaching agreements in principle with Synopsys, Viewlogic and CADIS (now part of Synopsys) for free use of their tools.

3.3.3 **Lessons Learned and Issues**. The assumption that tool suppliers could be persuaded to provide free tools and accept a minimum payment for maintenance was false. Negotiations have been difficult and caused delays in schedule. Adequate funding for tools and pre-arranged agreements with tool suppliers are necessary even when suppliers will benefit from tool development. The current tool plan for RASSP is to deal with each supplier individually. Those that fully support RASSP and RASSP goals and directions will continue to supply their software for free or nearly so. In other cases the software will be available to RASSP on a maintenance only arrangement. A third scenario is to have short term rentals for tools that have been identified as a requirement for one of the RASSP teams. If these methods do not work, tools that can be transferred from within a RASSP member company to the RASSP Project will be sought. Finally, if a tool is necessary yet not available by any of the methods outlined above, a RASSP member company or companies will purchase the tool.

3.3.4 **Future Activities**. Tool evaluations and acquisitions will continue. Four tools will be evaluated shortly: Galaxy, InQuisiX, SES, N!Power, Rippen. These tools were either recommended by the Systems Engineering team or the RDE team. In the normal course of development of the RDE or of the RASSP process definition, there are needs or enhancements that need to be filled. Exposure to new tools which could potentially fill voids are initially identified and then given a priority for evaluation.

TABLE 3-1
DOCUMENTED IDE TOOLS

TOOL NAME	MANUFACTURER	DESCRIPTION
Netmaker	Aggregate Comp.	Software Development
Aptix	Aptix	Reconfigurable Wiring
Autocad	Autodesk	3D Mechanical Engineering
AutoTherm	Mentor	Thermal Mechanical Analysis
CodeCenter	Centerline	C Code Development
db/dbxtool	UNIX™	C Debuggers
Falcon	Mentor	ECAD Framework
FrameMaker	FrameTech	Word Processing
gcc	GNU	C/C++ Compiler
gprof	GNU	Software Profiling Tool
Macproject	Claris	Project Management
Make	UNIX™	Software Development
MATLAB	Mathworks	Algorithm Development
Mentor ECAD Sys	Mentor	Schematic Capture/Layout/Simulation
ObjectCenter	CenterLine	C++ Code Development
Opnet	MIL3, Inc.	Network Performance Modeling
pSpice	MicroSim	Analog Simulation
Purify	Pure Software	Software Leak Detection
Quad Design	Mentor	Signal Integrity Analysis
Sentinel	Sentinel	Software Leak Detection
SL-GMS	SL Corp	OO Graphical Modeling System
SCCS	UNIX™	Source Code Control System
SPW/CGS	Comdisco	Algorithm Development
Sun cc	Sun	C Compiler
Sun CC	Sun	C++ Compiler
Synopsys	Synopsys	VHDL Development Environment
TDS Wavemaker	TSSI	Waveform Test Vector Generation
TDS Software	TSSI	Sys Design Verification/Analysis/Test
Ptolemy	UC Berkeley	Algorithm Development
VHDL Simulator	Vantage	VHDL Simulator
XDesigner	VI Corp	X GUI Builder
PowerView	Viewlogic	Schematic Capture/Layout/ Simulation
VxWorks	Wind River	Real-time Operating System
Mathematica	Wolfram	Algorithm Development
Xact/Xilinx	Xilinx	FPGA Designer

3.4 NEW TECHNOLOGY EVALUATIONS

3.4.1 Goal. To locate and assess candidate technologies to fill voids in the design methodology.

3.4.2 Status. The RDE Team continuously monitored the Technology Base Broad Area Award contractors in key areas including communications, management, design automation, microelectronics, packaging, signal processing architectures and software. White papers were prepared on new technologies in microelectronics, software, design automation and databases.

3.4.3 Lessons Learned and Issues. An unexpectedly large amount of resources will be needed to adequately track, interact with, and evaluate technologies being developed under RASSP's BAA contracts. The RASSP team has recently realigned its approach to include a "one-on-one" engineer responsible for each contract.

3.4.4 Future Activities. Three technologies will be evaluated: Prototype Systems, DFM-Stanford, Regression tests.

3.5 COST MODELS

3.5.1 Goal. To develop accurate reliable cost estimation models for hardware and software.

3.5.2 Status. A final report was generated describing both the Hardware and Software Estimation Models. The software model utilizes the SEER estimation model. Motorola did an extensive evaluation of available models, including the PRICE model, and judged SEER model to be superior. Motorola has a site license for the hardware and software modules (there is one other module for ASIC design which the RDE does not have).

3.5.3 Lessons Learned and Issues. The principal lesson learned is the importance of working closely with the hardware and software developers. These individuals can provide needed insights and guidance (e.g. "code reuse" or "hardware design reuse" factors to apply). Greater understanding yields a better, more accurate product.

The principal issue to be addressed is that the SEER model only runs on PC and MAC platforms, making automatic metric collection more difficult. Other platforms must be assessed.

3.5.4 Future Activities. SEER modules for life cycle cost and software sizing will be investigated.

3.6 INTEGRATION

- Extensive software evaluations revealed that tools for integrating the product development environment are not mature. Multiple unconnected design frameworks and point solutions characterize the current Signal Processor design process.
- Since enterprise level integration tools are not mature, it is not yet possible to meet the RASSP objective of using COTS tools. This will be possible only if suppliers adopt RASSP's integrated product development solutions. The RDE team utilizes applicable standards and standard interfaces wherever possible. Other interfaces developed by the RDE team will be submitted to CFI for adoption. Incorporation of new application programs that have these standard interfaces will allow for easy integration.

3.7 DESIGN ENVIRONMENT SUPPORT (DATABASES AND NETWORK COMMUNICATIONS)

3.7.1 Goals

- To define and implement the RASSP Engineering Database and to Implement a standard interface to the Product Design Manager (PDM).
- To study, define and implement the network architecture needed to support the distributed development environment. Specifically:
  - » *Program Year 1 and 2* - Build essential network services to allow smooth operation of the Lockheed Sanders team.
  - » *Program Year 3 and 4* - Migrate these technologies into the RDE releases, and investigate other technologies for use within the Lockheed Sanders team.

3.7.2 **Status**. The manufacturing interface for the Motorola printed wiring board facility has been identified. It is a Mitron CIMBridge interface from Allegro to the manufacturing facility. Motorola also has purchased a Mentor interface but has yet to install and customize it. A schedule is being developed to provide the Mentor capability by October, 1994 so that the Demonstration Team can fabricate boards.

The updated requirements for secure data sharing and conferencing have been completed. Commercial encryption and decryption products will be evaluated ..

A RASSP network road map was completed and the Manufacturing Interface Data Translator Task started (e.g., IGES and PDES). This interface will continue to be developed to allow close coupling between the design and manufacturing environments, especially for printed wire boards and multichip modules.

3.7.3 Lessons Learned and Issues

- Originally, it was thought that Sherpa would be the foundation for the Enterprise Framework, and that flow management would be added to Sherpa product. However, the Enterprise Framework is in fact a set of cooperating services which the Sherpa architecture cannot support. Therefore, a standard PDM interface rather than a Sherpa solution was selected to promote "plug and play" with a variety of PDMs.
- With respect to network technologies, screen-sharing capabilities could be powerful tools for remote interaction. These capabilities were believed to be available in useful implementations. However, the commercial implementations (particularly Sun "ShowMe" and Farallon "Timbuktu Pro" for Macintosh) are immature. The evaluation of commercial screen-sharing products will continue.
- It was also believed that one of the various mechanisms for secure data protection (Kerberos, application safeguards, software encryption, or hardware encryption) would provide a secure RASSP network utilizing public networks as a foundation. However, each of these candidates has a shortcomings. A complete solution will probably evolve in the next set of Internet Task Force standards.

3.7.4 Future Activities

- Implementation of a PDM-independent interface in RDE 1.0.
- Mapping of the Demonstration Team product structure to the Demonstration Team process.
- Full implementation of the distributed network architecture. This effort will include integrating SCCS and FTP, establishing shared file systems across the Lockheed Sanders Team, continued adoption of audio/video conferencing, evaluating of screen-sharing technologies, and implementing an encryption mechanism to protect electronic information

4.0 RASSP DEMONSTRATION TEAM ACTIVITIES

4.1 GOAL

To validate the RASSP Process and RASSP Design Environment (RDE) in the context of real-world signal processor design. Over the course of the RASSP Program, three full releases the RDE will be used, along with corresponding releases of the RASSP Process. Each release will be used to develop model year upgrades to the demonstration vehicle. The specific objectives of this task are to:

- Use an embedded signal processing system as a test case, spanning the development cycle from concept through specification, architecture analysis and design, to manufacture and support, so that the entire RASSP process can be evaluated as it evolves during the contract.

- Design the system using the RASSP model year concept; i.e., upgrade the system design rapidly and often, incorporating the latest technology and incrementally upgrading the system throughout its life cycle.
- Provide process metrics and lessons learned for methodology and process refinement. Measure the progress toward reducing product development time by a factor of four.
- Provide feedback on the usefulness of specific tools and the design environment.
- Provide clear, convincing data that RASSP methodology is practical and effective for complex design tasks.

This report addresses Model Year 0. This work was performed primarily with tools from the RDE tool set. RDE integration or infrastructure capabilities were initially not available, although late in Model Year 0, RDE 0.1 became available and was beginning to be used by the Demonstration Team.

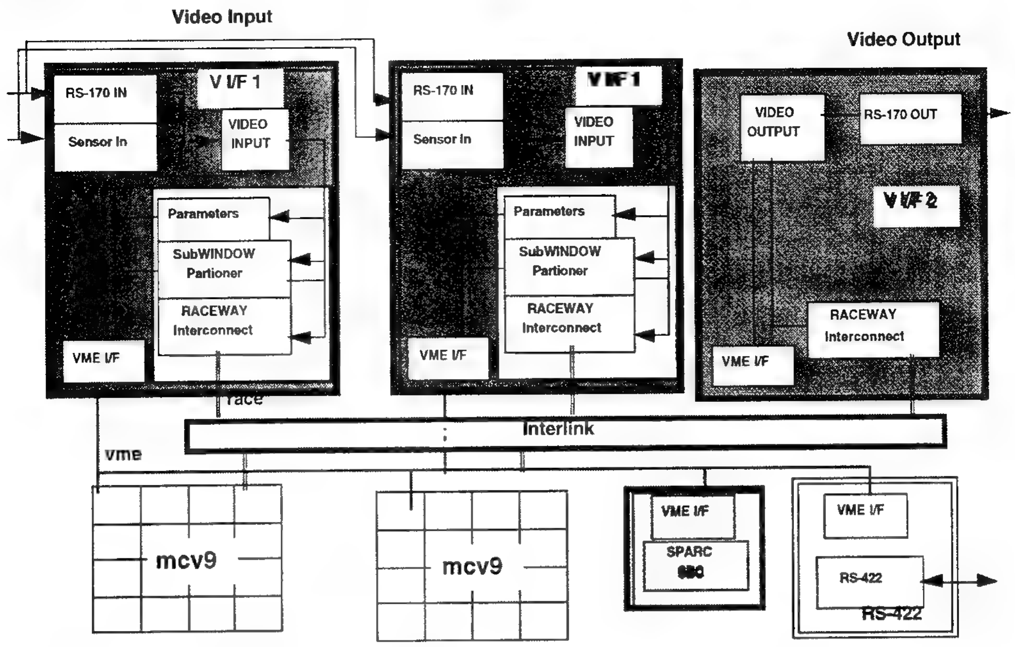
4.2 APPROACH

4.2.1 Overview. The selected demonstration vehicle is an airborne infrared search and track (IRST) processing system using programmable processors (Figure 4-1). This system takes sensor or RS-170 video data's input, detects objects of interests, tracks them and output the results as RS-170 video. It presently is planned to (leave room for other architectures in later iterations) uses a heterogeneous Multiple Instruction Multiple Data architecture with commercial off the shelf processor chips, operating systems and system software tools.

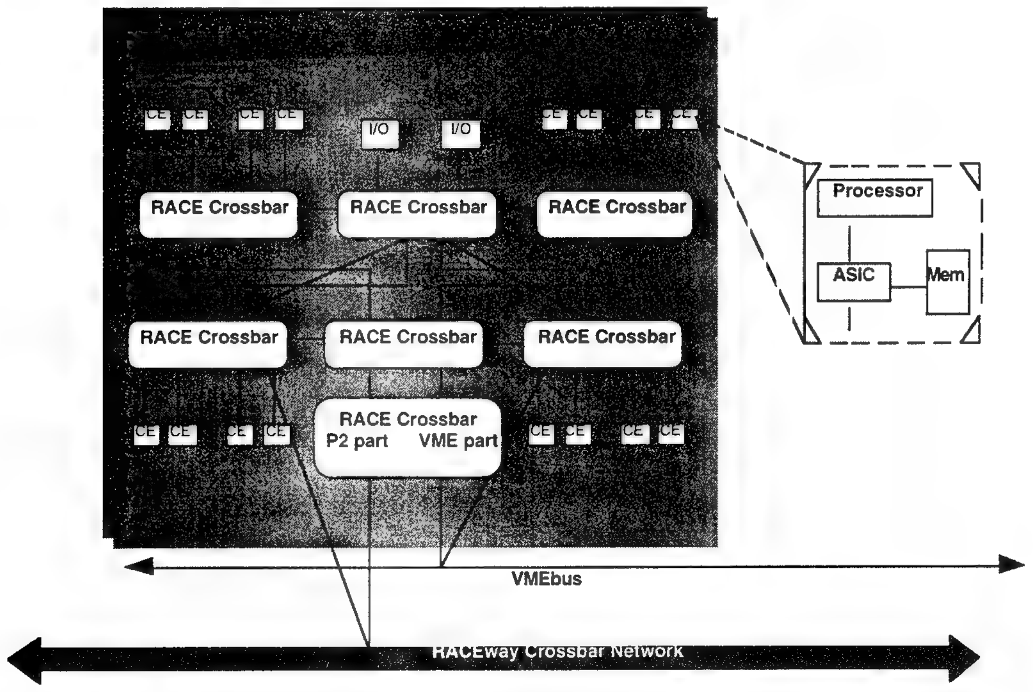
The IRST processor was selected for six principal reasons:

- *Scalability* - IRST algorithms were available in a scaleable manner for a coarse grain Multiple Instruction Multiple Data parallel processor.
- *Modularity* - IRST algorithms and software are modular, which allows RASSP process exploration at different levels of rigor for different functions.
- *Tools.* - Some algorithms are described at the math level in Matlab, one of the RASSP analysis tools, thus allowing top-down exploration of hardware/software partitioning and design.
- *Ease of implementation* - Some existing algorithms easily lend themselves to hardware implementation (e.g., convolution and registration).

Model year 0 provides a complete design cycle from requirements capture to hardware and software integration. Algorithms and some C code provided a starting point for the design. All new code is being developed in Ada. As an expediency, the core algorithmic code that was available in C, has been encapsulated in Ada, to enhance its maintainability, and compatibility with the rest of the Ada code. The multiprocessor system design was selected as a result of performance level modeling that compared three alternate architectures. Use of Commercial Off The Shelf components has been maximized. Only the interface cards are unique construction since no commercial modules could be found that operate to the required 135 MB per second sensor rates.



4-1a. IRST System Hardware



4-1b. IRST System hardware MCV9 Details

Figure 4-1. Selected Demonstration Vehicle

The Demonstration Team focused primarily on the top-down VHDL design phases of the Model Year. Architecture trade studies started in Fall, 1993. Performance level modeling of three alternate 80 processing element MIMD processor architectures was completed by February, 1994. This resulted in the selection of a multiple-i860 architecture using a 160-MBS interconnect network. Software partitioning was modeled in the performance level simulator.

Specific hardware/software issues were uncovered during the initial modeling and subsequent refinement. The most significant issue is how to input large infrared images at 135 Mbytes/second. Commercial, off-the-shelf solutions alone were not able to meet the requirements.

Requirements and high level designs were generated for the software architecture and data input/distribution card. Domain analyses were done to facilitate reuse in subsequent model years. Top down refinement of the design was captured in VHDL with bus interface, functional, and component level descriptions.

Currently, integration of the hardware and software on the virtual prototype is underway. Upon completion, physical design of the modules will be completed, boards fabricated, assembled, and checked-out. The schedule is to complete the integration of hardware and software by January 1995.

4.2.2 **Demonstration Process and the RDE**. The Demonstration Team is using key elements of the RDE to implement the RASSP process:

- *Top Down Design Refinement* - Work is released for lower level design before the current level of definition is complete. This allows rapid design but requires good coordination when data is passed forward and returned with implementation constraints. Incomplete data is marked with level of completeness. This path is utilized in order to achieve a good design as rapidly as possible -- where a good design is one which meets all the requirements, is manufacturable, and does not break the bank with development of new components.
- *Requirements Capture Using Domain Analysis* - Domain analysis helps build reusable elements and evaluate the potential performance of building blocks from the library. It establishes requirements and partitioning that support design reuse for subsequent model years.
- *Design Trades* - Tradeoffs assess implementation alternatives that meet design requirements. Reuse libraries containing Ada and VHDL influence the design trades.
- *Standard Language* - VHDL and Ada support reuse and provide the long term stability of a standard language, which will permit parts replacement long into the future.
- *Simulation* - As trade analyses are completed, performance level is simulated to ensure system timing is viable. The team progressively refines the simulations to define and check interfaces, functions, and interconnects of components. Parts libraries are augmented with new parts as needed. The simulation tasks build a complete hardware and software virtual prototype of the processing system.

- Virtual Prototype - The virtual prototype is the climax of the first phase of the development cycle. It confirms that design decisions are viable and customer needs are met. It tests the hardware and software design elements together, reducing the number of errors at this interface.
- Build - After complete checkout of the virtual prototype, the hardware design goes to physical design, fabrication, rapid parts procurement, and checkout. Software design changes are finalized. Hardware and software are integrated in the laboratory and then in the final system.
- Model Year Upgrade - The state-of-the-shelf design replaces the state-of-the-art design. Building the system with available technology from reuse libraries, current vendor parts, portable software elements, and scaleable multiprocessor architectures accelerates the design cycle and cuts the design cost without precluding the later inexpensive introduction of more advanced parts as they become readily available.

4.2.3 RASSP Improvements to the Product Development Process. The product development process being used for the Model Year 0 demonstration is shown as Figure 4-2. The difference between this process and current industry practice is the introduction of a virtual prototype; i.e., a set of models in VHDL used to simulate system behavior. Use of the virtual prototype is to move validation of system behavior and interfaces from the integration, test and manufacturing phase to the system/subsystem design phase.

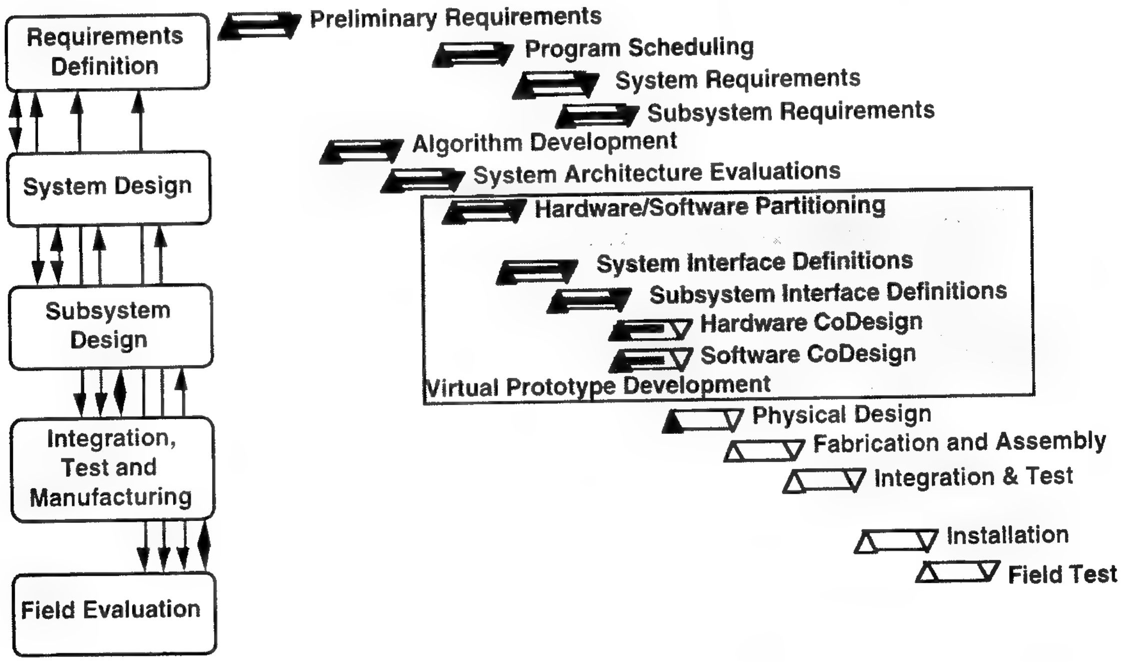


Figure 4-2. Demonstration Team Product Development Process

The early on check out of system behavior by using the virtual prototype should result in reducing the time spent in integration and test. This expectation of time reduction in the integration and test is founded on three concepts; better understanding of expected system behavior, fewer iterations in the rework cycle, greater visibility into the interactions of system components.

Better understanding of expected system behavior is achieved, in part, through early discovery of flaws in the communication between hardware designers and software developers. These flaws in their communication are exposed through use of the virtual prototype. It allows both groups to observe model behavior and compare that behavior to their expectations. When the expected behavior is different from the model's behavior, the hardware designers and software developers can resolve the differences and then refine the model to better perform as expected.

The fewer iterations in the design cycle concept is the usage of synthesizable models for the non commercial, off the shelf functions in the virtual prototype. The synthesizable models can be converted to a physical design, an application specific integrated circuit), a field programmable gate array, even a printed wiring board and their behavior can be compared to the original synthesizable model to validate correct implementation.

Synthesized function characteristics not available at development of the synthesizable model can be added. This allows accounting for the physical design effects in the system design which in turn means fewer discoveries of unexpected behaviors in the integration and test phase and less rework.

Greater visibility into system component interactions results from the virtual prototype's simulation environment. During model development, desirable internal state values can be brought to the model periphery for monitoring. Also, during simulation of the virtual prototype, non physical internal values can be probed for current values. Lastly, complicated test generation and comparison are accommodated in the modeling environment using test benches and the richness of VHDL.

4.2.4 **Virtual Prototype Development**. Using the virtual prototype during the design phase (Figure 4-3) is expected to reduce integration, test manufacturing time. This is a result of reducing the risk of failures in interfaces between subsystems by simulating them before fabrication. The interfaces are simulated from the application software through the interface driver software and subsystem interfaces and functions to external system interfaces. This reduction in time and effort in the integration, test manufacturing phase is offset by the larger effort in the design phase spent modeling and testing the model of the design.

The development process comprises three phases:

- *Requirements Definition* - The system to be modeled is defined in terms of the system interfaces, system function, subsystems and subsystem interfaces, subsystems includes hardware and software. Identification of any embedded processors and interface driver software and their functions are also done at this phase. Unknowns are "stubbed" inside a "box" and flagged for later development.

- Model Development Phase - The system and subsystems are modeled in terms of bus interface and behavior models. Bus interface modules provide all interface signals, timing, and function, but no internal detail. Behavior models extend the interface module with the internal functional detail inside the module.
- Integration Phase - The system and subsystems are simulated to determine correct operation. In the subsystem with the embedded processor this is done in three phases to progressively checkout the system from building blocks to the full system:
  - » Phase 1 - Verification that the ISA and the interface driver software perform as expected.
  - » Phase 2 - Verification that the interface driver software running on the ISA within the subsystem performs as expected.
  - » Phase 3 - Simulation of the entire system to verify that the software running on the ISA provides the required communication between the application software and the system's external interfaces.

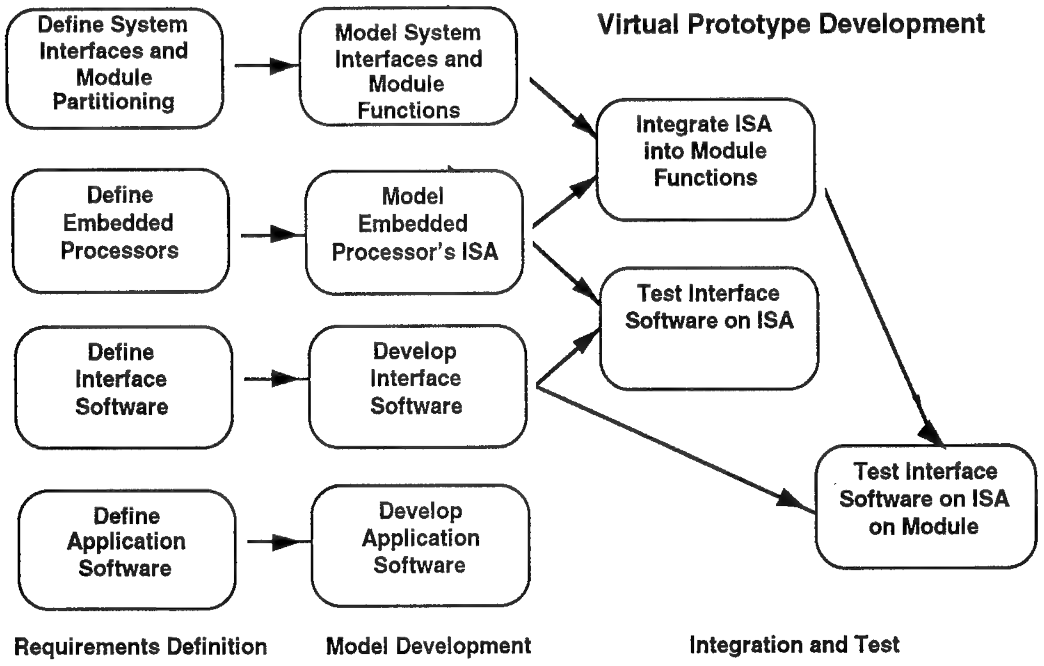

Virtual Prototype Development
Define System Interfaces and Module Partitioning
Model System Interfaces and Module Functions
Define Embedded Processors
Model Embedded Processor's ISA
Define Interface Software
Develop Interface Software
Define Application Software
Develop Application Software
Integrate ISA into Module Functions
Test Interface Software on ISA
Test Interface Software on ISA on Module
Requirements Definition
Model Development
Integration and Test

Figure 4-3. Visual Prototype Development Process

4.3 STATUS

4.3.1 **Architecture Selection**. An initial architecture classification method (Table 4-1) was defined early in the Demonstration project to allow both tracking of alternate architectures and to support architecture selection during the design process. This classification method has been used to communicate which types of architectures have been explored and how well they met specific application requirements. It has also been used to indicate characteristics of a particular architecture class and how well it supports candidate applications. In addition, the classification method has been used to illustrate the scope of alternative

TABLE 4-1
ARCHITECTURE CLASSIFICATIONS

	PROGRAMMING MODEL
a	SIMD
b	MIMD
c	SPMD
d	Data Flow
e	Data parallel
f	Message Passing
f1	Asynchronous
f2	Synchronous
f3	Broadcast
f4	Multicast
g	Shared Memory
g1	Non Uniform Memory Access (NUMA)
g2	HW Cache coherence
h	Homogeneous/heterogeneous
	INTERCONNECT
a	Mesh (dimensionality)
b	Tree (dimensionality)
c	Cross bar (dimensionality)
d	Busses (number of)
e	Point to point
h	Clustering
	METRICS FOR EVALUATING:
a	Hops/processor cluster
b	Max latency in hops
c	Node bandwidth
d	Bisection bandwidth
e	Topology limits
f	Allowed number of breaks in topology
g	Processor independent interconnect
h	Deterministic throughput
i	Interconnect power/processor node
j	Interconnect weight/processor node
k	Interconnect volume/processor node

	PROCESSOR
	PROCESSOR GEOMETRIES
a	Single node data processor
b	Single node DSP
c	Multinode data processor
d	Mult node DSP
e	Number of Cache Levels
f	Vector processing elements
g	Proc elements per memory unit
h	Ports per memory unit
i	memory/CPU coupling
	METRICS FOR EVALUATING
a	MFLOPS/watt
b	MFLOPS/dollars
c	MFLOPS/volume
d	Optimized for algorithm functions
e	MEMORY unit bandwidth (MB/S)
	SYSTEM SOFTWARE
a	C
b	C++
c	Ada
d	Parallel processing language
e	Multilevel secure OS
f	Multiprocessor debugger
g	Profiler
	METRICS FOR SELECTING
a	Compiler availability
b	Compiler efficiency
c	OS size
	CHARACTERIZATION
a	Packaging levels
b	Thermal Design

architectures considered during the tradeoff process. Note that this classification taxonomy is only a communication aid. The classification taxonomy was not intended to drive the design process. Note that the RASSP System Engineering activity has described a more formal architecture classification approach in recent months. This approach will be used by the demonstration activity in future architecture classification/selection work.

4.3.2 **Performance Model Development**. Performance level modeling is used to select the system architecture best suited to perform the required task.

To reduce the time and resources needed to generate and simulate a performance model, a subset of the larger system can be modeled that provides results that represent the entire system. Once the design iteration cycles are complete and the optimal architecture identified, a complete optimal architecture model of the can be generated to provide absolute performance statistics. This is the process the Demonstration team used in performance level modeling of IRST architectures (VHDL was the simulation medium). Performance Level Modeling libraries derived from the Air Force / Honeywell GPD contract, provided the infrastructure for the modeling effort.

The Demonstration team modeled and simulated a Mercury MCV9 board. Each MCV9 board using the GPD performance level modeling primitives contains 16 i860 processors. The simulation job required over eight hours to generate three seconds of simulation time.

Elaboration of the 32-node and 80-node architecture could not proceed with current RAM resources (Sparc 10s with 128 MB). Compiling the 16-node architecture took about 10 minutes and elaboration approximately 15 minutes. Simulation time through initialization required about 15 minutes.

While the entire system could not be simulated, sufficient portions were analyzed to show the feasibility of our chosen architecture. The decision was to extrapolate results from the system subset simulation to the entire system, proceed with system requirements development and design, and return to the system performance model as performance issues arise.

4.3.3 MCV9 and I860 Processor ISA Model Development

4.3.3.1 ***ISA Processor Model Development*** Later, a more detailed model of the MCV9 was generated. This included modeling the Instruction Set Architecture (ISA) of the i860. This represents all features that the programmer can touch, e.g., register, instructions, and memory. This simulator can run actual assembly, C, or Ada code. The current I860 model consists of instruction decoding and execution with Bus Interface behavioral modeling and memory reads and writes. The following approach is used to date.

1. The internal register files (integer and floating point) are modeled using integer arrays and all other registers are either integer or BIT_VECTORS depending on their use. All internal registers are represented using the VHDL variable data type.
2. The instructions are decoded based on the opcode of each instruction and a tree structure is used to further decode the instruction when necessary.
3. Pipelines are modeled using variables that store the operands of a given instruction until the last stage where the execution is performed. The size of the pipeline is based on the functional unit and precision of the operands. The current approach is faithful to the i860.
4. The memory was modeled using an internal memory model with two 4096 arrays of integers. The memory was loaded from files using the Styx C interface.
5. The instructions are executed using procedures written to perform the respective functionality. This functionality is contained in a separate package and is reusable across processors with similar functional units.

6. Currently the floating point single and double precision are implemented using the Mathstyx package of vantage but future versions will implement the IEEE 754 standard for both precisions with a cost in model execution speed. Single precision floating point add and multiply are currently under development and will be contained in a math package called IEEE_754.
7. The bus interface model implements single and burst reads and writes to memory. The model characterizes all processor signals with timing. Protocol function is modeled. Internal processor details are not modeled in the interface model. Pipelined and I/O reads , and interrupt functionality must be implemented.
8. Trap and reset handling functionality are about 50% complete. This includes all exception handling on floating point instructions, address misalignment, reset, and instruction faults.

4.3.3.1.1 ***Instruction Testing*** Instructions were tested after the functional code was written. Small test routines were used that loaded the appropriate registers with data and performed the specified operation (Figure 4-4 shows an example of this). This method was used until the compiled code from an i860 compiler was available. The compiler, depending on the application, does not utilize all the instructions so the small test method still is needed to verify the less used instructions.

```
0                                   -- First address of instruction memory

22                                  -- Last address of instruction memory

00010100001000100000000000000001    -- LOAD_32_i R2 <- mem(@R1 + 0)
00010100001000110000000000000101    -- LOAD_32_i R3 <- mem(@R1 + 4)
10010000010001000001100000000000    -- ADDS R4 <- @R2 + @R3
00010000010001010001100000000001    -- LOAD_32 R5 <- mem(@R2 + @R3)
00010100100001100000000000000101    -- LOAD_32_i R6 <- mem(@R4 + 4)
00010000010001110001100000000000    -- LOAD_16 R7 <- mem(@R2 + @R3)
00010100100010000000000000000010    -- LOAD_16_i R8 <- mem(@R4 + 2)
00000000010010010001100000000000    -- LOAD_8 R9 <- mem(@R2 + @R3)
00000100001010100000000000001101    -- LOAD_8_i R10<- mem(@R1 + 13)
00001100000000000101000000000010    -- STORE_8 mem(@R0 + 2) <- @R10
00011100100111110010111111111100    -- STORE_16 mem(@R4 - 4) <- @R5
00011100100111110011111111111001    -- STORE_32 mem(@R4 - 8) <- @R7
00110000101010110000000000000000    -- LOAD_C R11 <- EPSR
00001000000000001001010000000000    -- IXFR  F1 <- R5
00111000001000000011100000000000    -- STORE_C PSR <- R7
00010100001000100000000000010101    -- LOAD_32_i R2 <- mem(@R1 + 20)
00010100001000110000000000010101    -- LOAD_32_i R3 <- mem(@R1 + 20)
00100000011011000001000000000001    -- LD_FLOAT F12 <- mem(@R2 + @R3) ++ (64)
00100100001000100000000000001001    -- LD_FLOATI F2 <- mem(@R1 + 8) ++ (64)
00100100000001000000000000100101    -- LD_FLOATI F4 <- mem(@R0 + 32) ++ (128)
00101000001000100001000000000010    -- ST_FLOAT mem(@R1 + @R2) <- F2 (32)
00101111110001000000000000100101    -- ST_FLOATI mem(@R30 + 32) <- F4 ++ (128)
00001000000000001001010000000000    -- IXFR  F1 <- R5
```

Figure 4-4. Sample Test Code

4.3.3.1.2 ***Compiled Code Testing***. Two compiled C programs have run on the i860 model:

- *Convert Celsius To Fahrenheit.c* - Data is loaded into memory Internally, and the program converts Celsius temperatures to Fahrenheit and vice-versa.
- *Fir.c* - Data is loaded into memory externally and the program filters the data using a nine tap Fir filter.

Testing of Ada code is being performed now. During the initial development testing of the virtual prototype, an Ada program for Sobel processing was written and tried on the ISA simulator. However, difficulties with the Ada run time system and run time error checking complicated the checkout of the ISA simulation because loading the full Ada run-time system in the virtual prototype is not currently possible. The IRST code is Ada based and is being checked out currently on the virtual prototype, albeit with continuing difficulty as the generated code occasionally branches off to run time systems. We are working to get the process for using Ada to run more smoothly.

4.3.3.2 ***MCV9 Model Development***. Six models were received from Mercury and converted:

- Processor Element - An i860 bus exerciser and memory
- CE-ASIC - ASIC compute Element that containing glue logic and control.
- Peripheral Bus (PBUS) control logic
- VME master/slave control logic - Principally FPGAs and PAL logic
- Buffers and Viewlogic primitives.

Six models were built for the MCV9

- Interlink
- Interlink TestBench
- MCV9
- MCV9 TestBench
- MCV9/Interlink TestBench
- MCV9/VME TestBench

Most of the Mercury models were combined to create the MCV9 model. Using schematics and going through the same conversion process as previously mentioned the structure VHDL of the MCV9 was created. To simulate the i860 processor the ISA model from Georgia Technology Institute was incorporated into this model.

4.3.3.2.1 ***MCV9 Models***. To expedite development of this processor module, the Demonstration Team acquired all the necessary VHDL models used by Mercury Computer Company to design their product. Mercury did not use pure behavioral level VHDL for these models, but instead used a mixture of View Logic schematics with VHDL code segments. The models had to be converted to 1076 compliant VHDL to adhere to the elected VHDL tool suite. This was accomplished using:

- Export 1076 - Creates a structural representation of the Mercury design in VHDL by first transforming each schematic component into an entity that is instantiated into the design. All components are then connected using signal that represent the schematic structure.
- Vantage IR1.0 - Converts the Viewlogic VHDL code into VHDL that is compliant with VHDL 1076.

4.3.3.2.2 ***MCV9 Test Benches***. A testbench was created to test the MCV9 model. The ports of this model included the VME interface, RACEWAY, Interlink and Test port. Various simulations were run to test various sections of the hardware. To optimize simulations only those modes that were involved in the simulation were included into the configuration file, all others were left "open". Four simulations were ran:

- *Temperature Conversion Program* - This MVC9 simulation primarily included one instruction set architecture model executing a small program that converts Celsius temperature to Fahrenheit and *vice-versa.* This is the same program that Georgia Institute of Technology used as part of its simulation. This test case was used to create a method of establishing a semiautomated process of writing code and downloading an image that was compatible with the instruction set architecture model. Some script files were created to enhance this process.
- *FIR Filter* - This simulation was the same configurations as 1 but the instruction set architecture executed a more elaborate program. The intent of this simulation was to check out and validate more of the instruction set architecture's instruction and compare the results against a known source. In this case a very simple FIR filter was written in "C" and executed on a workstation. It was then compiled for the i860 and downloaded to the instruction set architecture model. A simulation was run using the same input data. Validation of the results was done two ways. One was to compare the output files from both tests. The other was using a Virtual Scope to display the output data. As an added check, the results were also checked against the output of MATLAB and displayed using the Virtual Scope. This was demonstrated at the RASSP Conference in August.
- *CE-ASIC Register Access* - This simulation was used to access and initialize certain control registers inside the CE-ASIC. This indicated that all of the appropriate signals for all of the models were initialized to the proper states during RESET.
- *Memory Access* - After initializing the CE-ASIC as mentioned above the instruction set architecture model was program to read and write memory.

4.3.3.2.3 ***Interlink***. The Demonstration Team also created an Interlink model. A TestBench for the Interlink was developed such that it included a master process connected to Port A and a slave process that was connected to Port B.

This was a stand alone model that could simulate data being passed from the Master process to the Slave process.

This simulation was used to test the RACEWAY interface. A testbench was created that instantiated an MCV9 and an Interlink model. The MCV9 was connected to Port C of the Interlink model. The instruction set architecture model executed code that passed data to the Interlink and the Interlink responded with the proper handshake. This included the instruction set architecture model, memory , one CE-ASIC, three XBAR ASICs and VME logic.

A testbench was create that instantiated an MCV9 and a VME driver model created by the Georgia Institute of Technology. Work is still in progress to test this interface.

4.3.3.2.4 ***Performance***. When the ISA model was run by itself (all other components "open") then the instruction set architecture could execute about 1200 instructions per second. As different models were added to the simulation (depending on if they were mixed-gate level or behavioral) then simulation times were increased and the effective instructions per second decreased. The worst case simulations thus far have included the MCV9 and the Interlink. Simulating the transfer of ten data words from the MCV9 to the Interlink took about 20 minutes .

4.3.4 Data Input and Distribution Module

4.3.4.1 ***Description***. The Data Input and Distribution Module (Figure 4-5) handles all IRST processor inputs and outputs. It accepts video data from either the AIRMS sensor or the RS170 Input Port, extracts relevant parts, and reformats it for transmission over the Mercury RACEWAY to the i860 processors. The Data Input and Distribution Module can also overlay processed video data with target symbology information, then convert the digital data to RS170 format for display. The Data Input and Distribution Module contains hard-wired logic and can be programmed via VME writes to control registers.

The Data Input and Distribution Module consists of three functional units (AIRMS daughtercard, RS170 Input Daughtercard, RS-170 Output Interface and RACEWAY Interface) interconnected via a Video Crossbar which routes video data in both point-to-point and broadcast mode.

4.3.4.2 ***RDE Tools Used***. The Data Input and Distribution Module was designed using the first iteration of the RASSP process. Table 4-2 lists the VHDL libraries and the tools used in the design process. The main design tool used was the Vantage VHDL simulator. Text editors, such as EMACS, were used to generate the VHDL. The VHDL was then compiled using the Vantage batch compiler, "analyze". The compiled designs were placed into a single library called "IRST". UNIX scripts were used to automate the compile process. Results were viewed using the Vantage "results display".

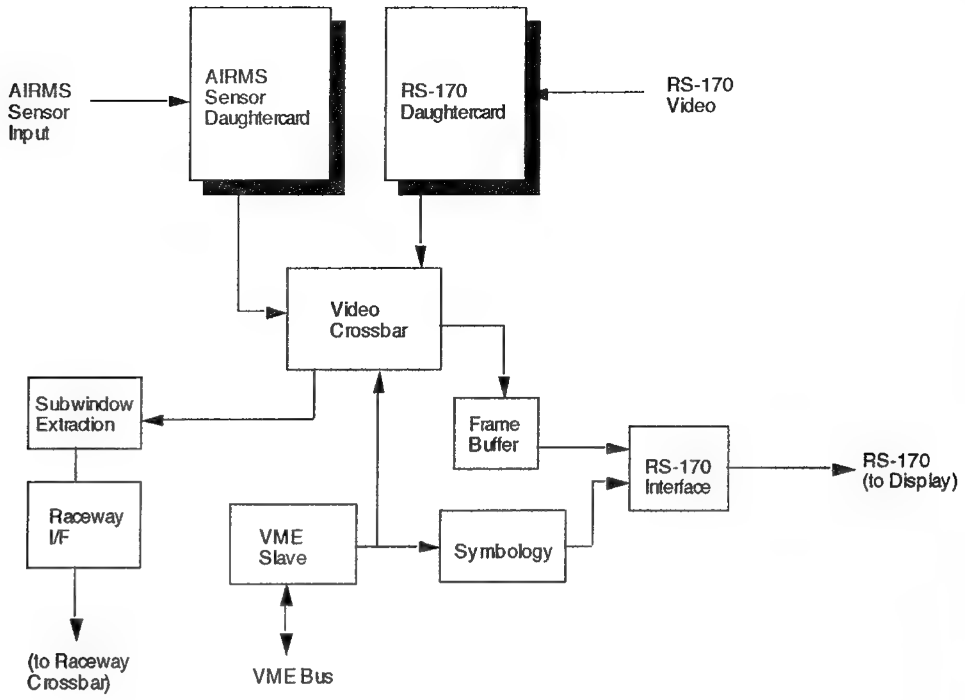


Figure 4-5. Data Input and Distribution Module

TABLE 4-2
DATA INPUT AND DISTRIBUTION MODULE DESIGN RESOURCES

VHDL LIBRARIES		
LIBRARY	**SOURCE**	**PURPOSE**
IRST	RASSP	Working Library
IEEE	Synopsys	Data Types, Data Conversion, Arithmetic
STD	Vantage	Text I/O

TOOLS	
TOOL	**PURPOSE**
EMACS	Text editor for VHDL generation
Vantage	VHDL simulation
Synopsys	Logic synthesis
XACT	FPGA routing
PowerPoint	VuGraf generation
MacDraw	Block diagrams
MS Word	Documentation
CuSeeMe	Video conferencing

Even though the Vantage VHDL simulator was used, the design was written using only standard IEEE 1076 VHDL. No vendor extensions were used. This allowed the VHDL code to be portable and run on other VHDL platforms. The only area where this caused a performance problem was in the used of TEXTIO instead of Vantage's Styx C interface for file input and output.

However, one deviation from the Vantage simulator was to use Synopsys' IEEE package instead of Vantage's IEEE package. Synopsys' package is "synthesizable" and allows the RTL code to be written for use by both Vantage and Synopsys. Since the Synopsys package is written using standard IEEE 1076 VHDL, Vantage compiled and simulated it without difficulty.

4.3.4.3 ***<u>Design Methodology</u>***. The Data Input and Distribution Module is a top-down design. Most of the module comprises digital logic, which can be modeled easily and accurately with VHDL. The analog portions of the RS170 Input and Output modules were modeled in VHDL at a behavior level of abstraction.

4.3.4.3.1 ***Requirements Analysis***. Model year 0 started with requirements capture and analysis. During the first month of this model year, requirements and high level analyses were completed. These included both the primary requirements for the IRST to fit on an AIRMS system and the derived requirements for parallel processing advanced IRST algorithms. Requirements included physical, test, performance, and derived requirements for hardware and software. Table 4-3 lists the manhours during this phase and lists the change rate of requirements over the first few months. Note June is the month that we held the preliminary design review for the hardware and software.

For portions of the derived requirements, where reuse looked likely e.g., video input function, domain analysis was performed to determine specific requirements that would meet F14, AIRMS, and other EO/IRST applications. The domain analysis resulted in a Data Input/Distribution function that should be reusable for many high bandwidth sensor inputs that need to be sent to parallel processing elements.

TABLE 4-3
REQUIREMENTS ANALYSIS METRICS

PERIOD	REQUIREMENTS	CHANGES	TIME (Days)
March, 1994	23	0	21
April, 1994	36	0	2
May, 1994	52	1	2
June, 1994	62	20	8
July, 1994	62	0	0

4.3.4.3.2 ***Bus Interface Level 1***. The first VHDL written was for a Bus Interface Description of the Data Input and Distribution Module. This produced an executable interface control specification/document which defined the Input and Output Pins for the Mercury RACEWAY, the VME bus, the AIRMS sensor, and RS170 Input and Output. Video Data was able to flow through one of the ports and out through another. No formatting of the data was done, instead the output data was the same as the input data.

The next step was to partition the Data Input and Distribution Module into 6 functional components: VME Interface, RACEWAY Interface, AIRMS Interface, RS170 Input Interface, RS170 Output Interface, and Video Crossbar. Bus

Interface VHDL models were developed for each of these functional units. This allowed the interfaces for each component to be defined. This was important, since three engineers were working on the Data Input and Distribution Module, and defining the interfaces early prevented misunderstandings later in the design cycle.

The Bus Interface Models consisted of VHDL entity descriptions and simple VHDL architecture models that allowed data to flow through.

Tables 4-4 and 4-5 list, respectively, the VHDL development metrics and total VHDL development.

TABLE 4-4
VHDL DEVELOPMENT METRICS

MODELING LEVEL	FILES	TIME (Days)
Bus Interface Level 1	34	27
Behavior Level 1	45	48
Bus Interface Level 2	7	3
Behavior Level 2	64	42

TABLE 4-5
TOTAL VHDL DEVELOPMENT

FILES	LOC	ENTITIES	ARCHITECTURE	CONFIGURATIONS	TEST BENCHES
160	9,761	49	69	16	19

4.3.4.3.3 ***Behavior Level 1***. Once Bus Interface models were developed for each functional component of the Data Input and Distribution Module, the models were refined into Behavior Level 1. This consisted of replacing the simple architecture models generated in the Bus Interface the unit. The individual Behavior Level 1 models were then connected to form an Behavior Level 1 model of the Data Input and Distribution Module.

At this stage, VME registers were defined, and commands could be downloaded over the VME to program the Video Card. Timing on the Behavior Level 1 VHDL was accurate to the clock level.

4.3.4.3.4 ***Bus Interface Level 2.*** Once the functional unit behaviors were defined, the units were decomposed into actual chip level components. Bus Interface Level 2 entailed defining the integrated circuits to be used in the design of the functional units, the building entity descriptions for these chips.

4.3.3.3.5 ***Behavior Level 2***. RTL descriptions were written for the five Field Programmable Gate Arrays in the design: RS170 Input, RS170 Output, Video Crossbar and Subwindow Extraction. The top level entity for each Field Programmable Gate Array was developed during the Bus Interface Level 2 stage, but the arrays were typically hierarchical and lower level entities and architectures were generated at this time.

The models for each functional unit were connected to form a Behavior Level 2 model of the entire Data Input and Distribution Module.

4.3.4.4 ***Modeling of Commercial Digital and Analog Parts***. The Data Input and Distribution Model uses many commercial off the shelf components: such as memories and FIFOs. Only at the Bus Interface HDL model. Generic memories and FIFOs were used before this time. For example, in the RS-170 Output interface, a RAM is used to store the video frame before it is output. In the Behavior Level 1 version, a single Dual Port RAM generic model was used, since it simplified the control logic. But in the Behavior Level 2 version, four IDT71024 128K by 8K random access memory models were used instead.

For each commercial device used, the entity was created using the pin names of the actual device. We used the information contained in the data sheet for the device to create a behavioral model of the device. In addition, the VHDL Generic construct was used to add timing to the models. Generics allowed changing of speed grades (e.g. from 12 ns to 15 ns) easily without modifying the model itself.

Generics were also used to specify the depth of the memory device. For example, the IDT72 family of FIFOs is used extensively in the design. Since we were using several depth sizes (from 256 to 4K), we created one behavioral model for the IDT72 and used generics to specify the one to use in the design.

The RS170 Input Module uses many commercial off the shell analog components. Each analog component was modeled with VHDL as an ideal mathematical behavioral model. As in the case with digital modeling, the appropriate data books were referenced to create a typical model. In addition, the ability to modify key parameters was added which allowed worst case simulation to be performed. The table below shows the analog components which were modeled.

4.3.4.5 ***Use of Test Benches***. Two types of test benches were used during the design process:

- *Type 1* - Tested single functional units in a controlled environment. Used in the initial mode development..
- *Type 2* - A model of the IRST system around the Data Input and Distribution Module. VHDL descriptions were written for the AIRMS sensor, RACEWAY Interlink, RS-170 Input Port, RS-170 Output Port and VME Master (see Table 4-6). This test bench was used to test the Behavior Level 1 and Behavior Level 2 models of the Data Input and Distribution Module.

TABLE 4-6
TEST BENCH DEVELOPMENTS

TEST BENCH	PURPOSE
RS-170 Input Port	Read image from and send to VDM over RS-170 input port.
RS-170 Output Port	Receive image from VDM over RS-170 output port and write to file.
AIRMS sensor	Read image from file and send to VDM over AIRMS interface.
RACEWAY Interlink	Receive image data from VDM over RACEWAY Interface.
VME Master	Send the VDM commands over the VME bus

4.3.4.6 ***Simulations Performed.*** The VHDL model for the Data Input and Distribution Module was simulated using both types of test benches. Test images were utilized to test the functionality of the module. The output images were compared with the input images to verify correctness. These tests confirmed that the data paths and protocols were correct.

Behavior Level 1 VHDL was used for performance level simulations of the Data Input and Distribution Module and the IRST processor. This simulation modeled video traffic produced by the Data Input and Distribution Module on the RACEway. The simulation took 10 hours on an HP 730 and simulated one frame of AIRMS data. The simulation results show that the RACEWAY is heavily utilized by AIRMS traffic for 69 ms, and then the AIRMS traffic will stop for 45 ms. This information was communicated to the software team as a guide to when to schedule non-input video traffic on the RACEWAY.

4.3.5 **IRST Application Software Development**. The IRST software comprises six processes.

- *System Controller* - Performs basic system management functions.
- *Data Input and Distribution Processor* - Receives video data from the video input cards and performs basic data formatting functions.
- *Image Scan Processor* - Receives the formatted video data from the VDP and performs the IR signal processing (detection) functions.
- *Track Processor* - Receives observations from the ISPs and performs the track functions.
- *Display Processor* - Receives track symbology information from the TP and receives video data from the ISPs and performs the video display functions
- *Host Processor* - Provides user command processing functions and system status display.

All these processes are portable and will run on any processor. The current baseline based on throughput analysis is to have the SC, VDP, ISP, TP, and DP targeted for the i860 processor, and the HP targeted for the SPARC 2 processor. The VDP and ISP are scaleable processes and will be replicated as much as possible for a given processor resource configuration. Figure 4-6 shows the allocation of software processes to the baseline hardware architecture.

Each of these processes is coded in Ada, with the exception that the inner algorithmic code in the Image Scan Process is C code that existed from other programs and has been encapsulated in Ada.

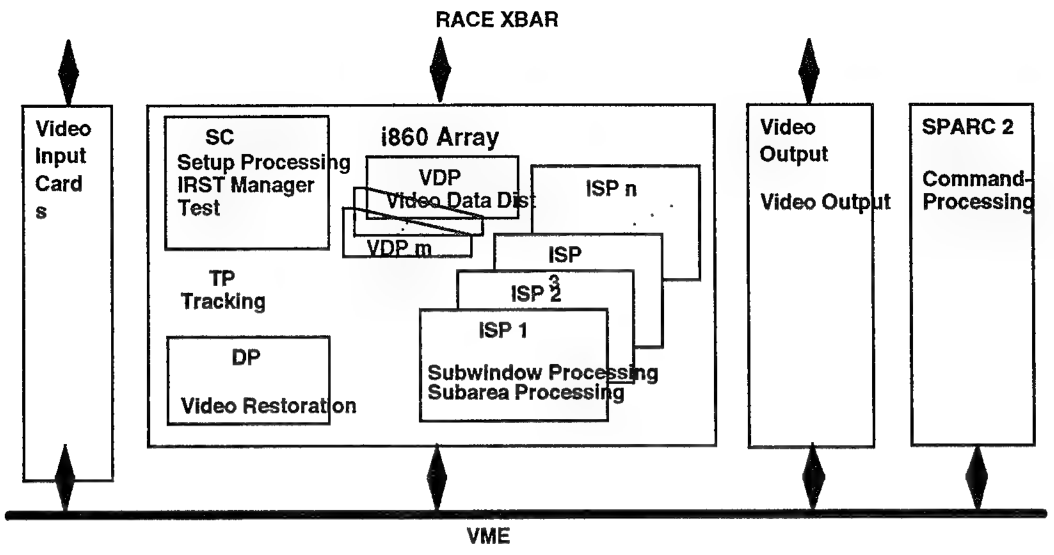


Figure 4-6. Software Allocation to Baseline Hardware

The *SYSTEM CONTROLLER SOFTWARE* performs architecture-wide management functions. Specifically, it:

- Performs the initial system startup by loading and starting all the i860-targeted processes.
- Is the focal point for all messages coming into the system.
- Receives health and status data from all other elements in the system and reports it to the user.
- Controls all system and software test functions.
- Executes the video input card control software.

The *DATA INPUT AND DISTRIBUTION PROCESSOR SOFTWARE (VDP)* performs video data formatting functions. The video input card routs the video data across the RACEway to the video distribution processors. There is one VDP per MCV9 processor card. Since the signal processing algorithms work on overlapping areas of the video data, the video distribution processors communicate the overlap regions to one another. The video distribution processors take the data from the video input card and the overlap regions and divides the data into subareas and forwards each subarea of data to an image scan processor. Since the AIRMS sensor provides video data in a column format and the image scan processor algorithms operate on row-formatted data, additional formatting is required before the data is sent to the image scan processor. When the VDP receives RS-170 format data from the video input card, the VDP has different functions to interpret the input, yet sends it to the image scan processors in the same row-organized format as required by the image scan processor interface.

The *IMAGE SCAN PROCESSOR SOFTWARE* performs detection processing on the IR video data. Observations are sent to the Track Processor for tracking, and the video data is sent to the Display Processor for video output display.

The *DISPLAY PROCESSOR SOFTWARE* takes the subareas of video data from the multiple image scan processors and reconstructs a full video image for display. Since the video output display is only about one-eighth the size of the AIRMS sensor field of view, the display processor can accept user commands to determine what portion of the sensor field of view is actually displayed. The Display Processor outputs the appropriate video to the video output card and sends the output card the appropriate control commands. The Display Processor software also receives the track information from the Track Processor, generates appropriate symbology, and overlays the symbology on the output video data sent to the video output card.

The *TRACK PROCESSOR SOFTWARE* performs tracking on the frame observations as reported by the image scan processors. It maintains track files on all current tracks.

The *HOST PROCESSOR SOFTWARE* provides the command and control interface to the rest of the system. It provides the interface to the AIRMS System Control Computer by receiving AIRMS commands and providing processor responses. Since the current baseline is for the Host Processor software to execute on a SPARC 2 processor which has a monitor, this software will also provide a user interface directly via the SPARC monitor and keyboard. In this way system commands and responses can originate from either an AIRMS operator or an IRST processor operator. The Host Processor software will also perform data logging and report writing functions.

4.3.6 <u>Summary</u>

- This interface software has been proven in 11 test cases on the Sun, with parameter variation (e.g., number of nodes, video frames, and pixel overlap) to develop "truth" expected results. These same tests executed on the Mercury (i860) development system using Ada driver routines produced results as the Sun algorithm development environment.
- The Data Input and Distribution Processor software was coded in Ada and unit tested. Test cases are being developed to exercise the video input card/video distribution processor/image scan processor interfaces and verify data flows.
- Prototype user interface displays to be driven by the Host Processor software have been developed. They are being evaluated by members of the RASSP team.
- Symbology generation routines were developed in Ada for the Display Processor software. Some unit testing has been done using a Sun workstation.
- The System Control and Track Processor software are under development in Ada.

4.4 LESSONS LEARNED AND ISSUES

- VHDL simulations can be extremely slow and require a lot of memory. It would be very beneficial to acquire state-of-the-art tools for fast VHDL simulations on a "super workstation" with 512+ MB of memory. This magnitude of memory is needed to run bigger VHDL simulations.
- The use of high fidelity models in simulations can cause many problems; e.g., the simulations with gate level or mixed gate level models. Not only do such simulations take longer, but systems resources can become limited. Some simulations produce 100-200 MB log files due to warning messages of the large scale integration components during RESET. Depending on the simulation, discretion should be used when turning this feature "off". Also batch simulations are more efficient for these types of simulations .
- For complex models such as instruction set architecture, incorporating visibility into certain parts of the model may increase simulation runs, but can prove invaluable during debugging.
- Whenever possible, type integer instead of std_logic_vector should be used. Type integer requires much less memory and simulator overhead.
- The capability of mixed mode (VHDL and Gates) simulation is essential to the success of "first time integration". The current plan is to use Mentor Quicksim (Version 8.0) combined with Smart Models to achieve this mixed mode capability. This appears to be a good plan, but all sites must have this capability for success.
- A better ability to forecast tool usage and acquisition. As the design evolves, all sites must have the ability to exchange data and the tools to support the development.
- VHDL itself does not completely guarantee portability modeling guides and coding conventions are required. For example, in our use of IKOS, Synopsys, and Vantage simulations for the performance level modeling, the VHDL code was originally developed using Vantage. We moved the code to the other two simulators. In moving to Synopsys, we found that the 32 bit internal data representations for time and various types, caused the code to not work.

  Another example were predefined types of milliwatts to gigawatts which could not be represented in Synopsys but could in Vantage. While this was not a show stopper (we do require that full range), it was an incompatibility between tools that took time to identify and fix. Other errors were found in differences between Synopsys and Vantage as to how they interpreted symbol overloading. All these problems were solved in portable manners but better guidelines will help us to avoid these problems in the future.
- Each site should have a contact person familiar with all aspects of the design. This way, information can be quickly shared and changed. Also, contact would always be available to help research information for the other sites.

- Lack of a Requirements Documentation Tool hampered system design. Powerpoint slides do not provided an adequate traceability path.
- When selecting commercial parts, ensure that a VHDL or Smart Model exists for the part. If no model available, allocate adequate time to make the VHDL model and verify that adequate means exists to test the model.

4.5 FUTURE ACTIVITIES

- Early Model Year 1 work will include acquiring an understanding of the F14 environment and existing IRST processor, developing a virtual prototype of the new RASSP processor based on existing Model Year 0 work, and development of the signal processing software necessary to implement the F14 IRST mission. This work is scheduled for completion by August, 1995, predicated on the amount and suitability of F14 documentation. Likewise, schedule revisions may be appropriate depending upon algorithm selections and availability of suitable code.
- Following completion of the Model Year 1 Virtual Prototype, hardware will be fabricated, assembled and tested. IRST software will be integrated. The integrated hardware and software will be tested in the laboratory and will be used to validate the Virtual Prototype. Additional development work will begin with respect to Model Year 2 requirements (an F14-capable system). This work is scheduled for completion around February, 1996.
- Model Year 2 hardware and software development, validated with a virtual prototype will be completed by August 1996. This task is expected to be similar in scope to that for Model Year 0 (1994) due to hardware and software changes. Software is expected to be ported to a new processing element. A heterogeneous processing system will be used with some fully programmable processing elements and some convolution elements for filtering. This work is scheduled for completion by February 1997.

5.0 <u>RASSP PROLIFERATION TEAM ACTIVITIES</u>

5.1 OBJECTIVES

The objectives of the RASSP Proliferation Team are to:

- Be the primary interface among the Lockheed-Sanders RASSP Development Team , RASSP User Community and Technology base.
- Educate the User Community on the RASSP concept, process and tools.
- Bring and support RASSP technology in the community.
- Obtain and forward community feedback to the RASSP Development Team.
- Promote RASSP commercialization.

5.2 APPROACH

The Proliferation Team established a communications infrastructure and personal contacts within the Development Team and the User Community. Specific actions included establishing:

- A Mosaic WWW server to support internal team and external communications of program plans and goals.
- An 800 number to support public enqueries about the RASSP Program
- An automatic FAX-back service on the 800 line to provide basic program material
- A set of E-Mail aliases and E-Mail access; e.g., *RASSP-INFO@RASSP.SANDERS.COM, RASSP-WWW@RASSP.SANDERS.COM.*

The Proliferation Team focused communication efforts at specific groups within the RASSP community; i.e., Tech-Base contractors, prospective Beta Sites, and the RASSP Educator/Facilitator team. These interactions were largely personal meetings, followed by significant phone and E-Mail contacts in preparation for activities in the coming years.

5.3 STATUS

The Proliferation Team's activities this past year encompassed five distinct areas:

- Interactions
- Standards
- Beta Sites
- Business Plans
- Commercialization.

5.3.1 <u>Interactions with RASSP Community</u>

5.3.1.1 ***Internal Communications***. During the first six months of the Program, the Proliferation Team helped establish and comply with the Electronic Control Room standards set by the RASSP Program Office. Team personnel participated in setting policies and standards for E-Mail transactions, especially those containing application files. A set of utilities and procedures were established for connectivity among diverse platforms (PCs, Macs, and UNIX systems).

To facilitate access to Program information both within and outside the RASSP Development Team, Proliferation personnel led the development of:

- An *Integrated Information Management Plan*, a draft of which is available on the World-Wide-Web server (described in Paragraph 5.3.1.4).

- A project wide address book containing the names and affiliations of RASSP personnel and key members of the Customer, Beta-Site, Educator/Facilitator, and Tech-Base communities. Entries include E-Mail and US mail addresses, and telephone, FAX and paging numbers.

5.3.1.2 ***Tech-Base Contractors***. Each Lockheed Sanders RASSP team leader and a number of company program managers attended the 1993 Tech-Base kickoff meeting in Washington DC. Many contractors there had existing relationships with one or more Lockheed Sanders team members. Almost half of the original Tech-Base contractors had relationships to the team as subcontractors, members of the Technology Review Board, or members of the SCRA Educator/Facilitator team. Official government contacts were identified at Wright Laboratories for each early Tech-Base contractors so that they would be kept in the loop with respect to interactions with their contractors.

Lockheed Sanders RASSP personnel have attended several Tech-Base meetings/reviews this past year.

Currently, a Tech-Base library is being established. In addition, each Tech Base contractor is being assigned a personal contact within the Lockheed Sanders team who will monitoring the contractor's progress, attend significant meetings, and providing input to Lockheed Sanders regarding technology insertion opportunities to be captured in the Program Roadmap.

5.3.1.3 ***RASSP Educator/Facilitator***. The Proliferation Team made contact with the SCRA RASSP Educator/Facilitator team early in the program; i.e., attending its kick-off meeting and coordinating a prior familiarization meeting where key RASSP personnel briefed SCRA on the program and in return were briefed on SCRA's program. This resulted in the identification of several areas for cooperation, in particular web technology where Lockheed Sanders will provide services for SCRA to use in its Initiative-Wide Web.

Other pertinent information provided to SCRA team were copies of Lockheed Sanders RASSP Visionary Demonstration and RASSP Process Document. The goal was to familiarize SCRA with Lockheed Sanders approach to achieving RASSP goals.

Further coordination meetings are planned for efforts involving newsletters, coordinated standards development, potential 800 number transfer to SCRA, and continued web activities.

5.3.1.4 ***World-Wide-Web Server.*** The Proliferation Team constructed and maintains a RASSP information World Wide Web (WWW) server available to the public at Universal Resource Locator *HTTP://RASSP.SANDERS.COM*. The server is based on Mosaic which has multimedia viewers for a number of platforms, Macintosh, PC and UNIX. Efforts are underway to integrate WWW into the Initiative-Wide web being constructed by SCRA. Services will be provided to SCRA to coordinate address-book, calendar, and document server capabilities so that information from individual contractors can be collated to the higher Initiative level.

5.3.1.5 ***800 Phone Number***. Lockheed Sanders maintains an 800 number (1-800-99-RASSP) to provide general team information. This number has an automated FAX-back service so that callers may obtain a printout of several general program information items. This service, though not broadly advertised, has received approximately 24 calls over the last six months.

Discussions with SCRA are underway regarding transfer of the 800 resource to SCRA for Initiative-Wide use.

5.3.1.6 ***Public E-Mail Access***. Several public E-Mail aliases were established and published for general use. These include a general information alias:

RASSP-INFO@RASSP.SANDERS.COM

and a special aliases for questions/comments generated directly from the RASSP Mosaic WWW server:

RASSP-WWW@RASSP.SANDERS.COM.

5.3.1.7 ***Design Automation Conference***. The Proliferation Team manned a both at the 3-day *31st Annual Design Automation Conference* in San Diego , California in June, 1994. The team gave Visionary Demonstrations given and distributed informational items .

5.3.1.8 ***Annual RASSP Conference***. The Proliferation team participated in the *First Annual RASSP Conference* held in Washington, DC in August, 1994. The team directly supported nine demonstrations of the Mosaic WWW RASSP server and the RASSP Visionary Demonstration.

5.3.1.9 ***RASSP Visionary Demonstration and Roadmap***. The Proliferation Team vigorously supported these tasks because they provide unequaled insight into the RASSP Program. In particular, the RASSP Visionary Demonstration has been well received by customer and other government agencies, potential Beta Sites, and SCRA team. It is a valuable proliferation tool, and will be enhanced by the scheduled integration of RDE - as built screens and functionality. A migration of the visionary demonstration from the Macintosh environment to an operational Sun environment will add to this value. Once operating within the Sun environment, it will be used to prototype envisioned RDE functionality and to incorporate evolving functionality as it becomes available through the RDE development effort.

5.3.1.10 ***Lessons Learned and Issues***. An unexpectedly large amount of resources will be needed to adequately track, interact with, and evaluate technologies being developed under RASSP's BAA contracts. The RASSP team has recently realigned its approach to include a "one-on-one" engineer responsible for each contract. The Proliferation Team will continue to coordinate these activities.

The wide acceptance and success of the RASSP web was a pleasant surprise. Initial release of the RASSP server met with 342 accesses the first week and has helped spread the notion of using this medium to communicate public information to both the Martin Marietta team and the SCRA REF team.

As expected, there is immense interest and support in the user community for the pay-per-use concept. A pleasant surprise came from our interactions with the vendor community over the first year. As described in detail in the first annual RASSP Business Plan deliverable, the third party vendor community is having problems with current licensing strategy. In brief, the support contracts purchased by users does not provide enough income to for both product support, FREE product upgrades, and R&D for new tools. These vendors were very interested in the pay-per-use approach as a means of supplying additional R&D income.

5.3.2 **Standards**. The Proliferation Team is tracking the 42 current standards for application to the RASSP Program. Table 5-1 lists these standards and their current status by category. A more complete "Standards Mapping" activity is scheduled for September, 1994. Discussions are underway with the Martin Marietta relative to tracking and supporting common standards of interest

5.3.3 Beta Sites

5.3.3.1 ***Beta Site Candidate and Selection Criteria***. Table 5-2 lists this past year's candidate Beta Site companies. These companies were given explanations of the RASSP and Beta Site program, and familiarized with the *Beta Site Selection Criteria* document. This publication defines Lockheed Sanders' requirements for Beta Site participation, and is a "checklist" to assess how well a specific project is meeting Beta Site requirements.

Two compiled C programs have run on the i860 model:

- *Convert Celsius To Fahrenheit.c* - Data is loaded into memory Internally, and the program converts Celsius temperatures to Fahrenheit and vice-versa.
- *Fir.c* - Data is loaded into memory externally and the program filters the data using a nine tap Fir filter.

The original plan was to release RASSP 0.1 software to an initial set of external Beta Sites in June, 1994. This proved to be an unrealistic goal. The software was delivered on time to the Program's own internal development team. However, the level of maturity and complexity that the software needed before it could be shown to have a positive "delta" on commercial organizations was severely underestimated. The consensus is that these attributes will not be available until release 1.0, scheduled for June, 1995. An alternative Beta Site strategy has been developed which is still aggressive, bit more supportable by the program. This strategy, presented to the customer at the 12 month review, includes the use of company internal and government laboratory Beta Sites in early spring of '95 as a means of refining the support material and mechanisms for the external Beta Sites scheduled for that summer. The successful use of RASSP by these internal projects and government labs should also help in establishing a "success story" for RASSP when talking to prospective Beta Sites. Use of pre-lease software by our own RASSP team (Demonstration and Benchmark groups) starting in the late fall '94 will provide critical feedback to the RDE development team as to functionality.

TABLE 5-1
POTENTIAL RASSP STANDARDS

TYPE	ACRONYM	CONTENT	STATUS
C		Domain specific message dictionary	
C	CCITT-4	Facsimile transmission format	
C	Compress	File compression	
C	DES	File encryption program	
C	FTP	File transfer protocol	
C	MS-?	Standard interface to DSP functionality in Windows	
C	TCP-IP	Network communications	
C	ToolTalk	inter-tool communications	available today/soon
DES	ISA	Instruction Set Architectures	
DES	JTAG	1149.1 - boundary scan	available now
DOM	AP203	Config. Mgmt of product data information	now?
DOM	CORBA	Common Object Request Broker Architecture	
DR	AP210	PCA information standard	emerging now
DR	AP211	Manufacturing Interface	
DR	DRPI	Design Representation - DRPI 1.3	available now
DR	EDIF 3.0	Electronic Data Interchange Format	available now
DR	IGES	Initial Graphics Exchange Specification	To be replaced by STEP
DR	Netlists	Netlists	available now
DR	OMF	Open Model Forum - simulator/model interface	in committee
DR	PCB	Printed Circuit Board	Draft in fall 1994
DR	Schematics	Schematics	available now
DR	Test		Draft in 1993, status unknown
E/C	TES	Tool Encapsulation	available now
EM	TSM	Task and Session Management	draft and pilot available
LAN	ADA 9x	ADA Language standard MIL-STD 1815	available now
LAN	BSDL	boundary scan description language	
LAN	C	the language	available now
LAN	Scheme	Extension Language	available now
LAN	SGML	Standard Generalized Markup Language	
LAN	Verilog	1364 - Verilog (but we aren't using Verilog)	1st ballot fall 1994
LAN	VHDL	1076-1993	new release available now
LAN	VITAL	VHDL modeling for sub-micron ASIC Libraries	in rework following first ballot
LAN	WAVES	Waveform and Vector Exchange Standard	available now
PDI	CDIF	CASE Data Interchange Format	
PDI	CIR	Component Information Representation	1995
PDI	EDB	Electronic Data Book	1995
PDI	EDI	Electronic Data Interchange Format	
PDI	LPM	Library of Parameterized Models	status
UI	COSE	desktop / workstation standards including:	
UI	HP-VUE	desktop or workspace manager	
UI	OSF Motif	GUI	
UI	X-11	X-window interface standard	R5

C	Communications	EM	Environment Management
DES	Design Standards	Lan	Language
DOM	Design Object Management	PDI	Program Data Interchange
DR	Design Representation	UI	User Interface
E/C	Encapsulation & Customization	WFM	Work Flow Managemen

TABLE 5-2
POTENTIAL BETA SITE CANDIDATES

• Raytheon	• Hughes Internal
• Digital Equipment Corporation	• ATT, Greensboro
• Loral (IBM Systems Div.)	• Woods Hole
• Lockheed - Ft Worth (JAFT)	• Army Research Lab (ARL)
• Lockheed - Ft Worth (F22)	• Honeywell, Albuquerque
• Motorola Internal	• Honeywell, Phoenix
• Lockheed Internal	

5.3.3.2 ***Lessons Learned and Issues***. As stated in Paragraph 5.3.3.2, an alternative Beta Site strategy has been developed which is more supportable by the program. This new material will include a detailed "roll-out" plan indicating what and when functionality will be provided by the RASSP system. The Proliferation IPPDT sees this as a key element in obtaining Beta Site "buy-in".

5.3.4 **Virtual RASSP Help Desk (RHD)**. Development and construction is underway on virtual RASSP Help Desk (RHD) to work in conjunction with the RDE problem reporting mechanism. Basically, the desk will:

- Provide information regarding process changes to help users adapt to them.
- Based on problem type, route reports or queries to the best source for response.
- Provide an intelligent front-end to problem reports filed in the past to help identify similar problems and provide solutions.
- Support real-time searching and viewing of documentation such as manuals, tutorials and video clips.

All interactions will be tracked and recorded for future access.

5.3.5 **Business Plan**. A draft Business Plan was submitted to the RASSP Program Office in late August for review and comment. The plan addresses business trends, value added by the RASSP process, barriers to RASSP and a market survey to determine RASSP potential.

The market survey is an essential tool for determining RASSP potential. Numerous questionnaires were sent out to technical leads of candidate projects. A preliminary analysis of the survey results was completed. At this time, however, not enough data regarding our RASSP process and product use and areas of leverage is available to generate a cost/payback model in sufficient detail to provide definitive financial benefit information. Future versions of the Business Plan are expected to include this level of information.

The process by which information is compiled on RASSP technology insertion candidates is continually reviewed and modified as needed.

5.4 FUTURE ACTIVITIES

Activities during the coming year will focus on continued:

- Interactions with the RASSP community, in particular the SCRA REF team and individual Tech-Base contractors.
- Interactions with and development of potential Beta Sites.
- Support of the RASSP system through development of support documentation, tutorials and help desk facilities.
- RASSP web server activities including integration of services with the SCRA REF Initiative-level web and supporting a multilevel access architecture.
- Continued commercialization efforts.

5.5 RELEVANT DOCUMENTATION

- *Beta Site Selection Criteria*
- *Beta Site Support Plan (Draft)*
- *1994 Annual Business Plan*
- *Visionary Demonstration V1.9*
- *Integrated Information Management Plan (Draft*
- LS RASSP World-Wide-Web Mosaic Server (URL http://rassp.sanders.com)
- LS RASSP Information Line and FAX-back service (1-800-99-RASSP)

6.0 TECHNICAL REVIEW BOARD

One of the more important elements of Lockheed Sanders program structure is the Technical Review Board which independently reviews program status. The Board's written report on RASSP submitted at the previous 6 month Technical Review has been used as one of the guiding documents for the past six months. The Board's report addressed nine issues (in italics):

1. *The focus on high-level process, methodology, and technical management issues is excellent. The problem is not just one of tool integration, but also one of coordination and communication among project engineers and managers.*

   LOCKHEED TEAM RESPONSE: This emphasis on issues is the foundation of Lockheed Sander' approach. It continued with additional focus in the last six months on the further development of Process and Methodology.

2. *The plan to maximize the use of network services in proliferation and in the RASSP development environment is excellent, not only because it will improve the productivity of this RASSP team, but also and more importantly, it will serve as a paragon for the process under development. Lockheed Team Approach*:

   LOCKHEED TEAM RESPONSE: The Team has continued to increase the capabilities of network services. In the last six months, new services have been implemented, including secure shared file systems across geographically distributed team members, an integrated FTP document repository using configuration management, and interactive World-Wide-Web applications. These capabilities are being used within the program and on other Lockheed contracts.

3. *The "hot mock-up" is an excellent concept. It helps not only convey how the tools should evolve, but also to develop the ideas by forcing concreteness.*

   LOCKHEED TEAM RESPONSE: We have continued using and developing the "Hot Mock Up" during the past six months. Three refinements have been implemented, with emphasis on clean-up and improving the concepts of project establishment and document promotion implementation.

4. *The emphasis of the work presented is on framework functionality; i.e., will it be up to the required tasks when the Lockheed team delivers their solution. It is possible that framework vendors will expand their frameworks to meet RASSP requirements. It is also possible that Enterprise framework suppliers will develop a suitable tool using various EDA frameworks. Can this be addressed?*

   LOCKHEED TEAM RESPONSE: The initial approach was to purchase a build or tool kit into which tools are integrated. This approach was abandoned when it became that an Enterprise Framework was needed to provide remote members of a team with services to cooperatively design and support signal processing systems. As described earlier in this report, after a series of evaluations, Cadence was selected to provide the framework.

5. *The team is relying on tool vendors for hardware and software synthesis, including partitioning and scheduling within a multiprocessor architecture. This, by contrast, is not being pursued as aggressively by the tool vendors. Is there a contingency plan?*

   LOCKHEED TEAM RESPONSE: The potential shortcomings of tools for hardware and software synthesis, and multiprocessor system partitioning and scheduling are being evaluated. The Team is convinced that tool vendors will not be able to deliver the capability required for typical signal processor problems. However, the Architecture Experiment (Paragraph) will determine if an object-oriented approach to signal processor software development can be defined, then extended to hardware development. This approach will be merged with some university tools (probably in 1996) for partitioning and scheduling.

6 *The tool integration problem is hard. The Lockheed team must acknowledge the diversity of models of computation in commercial tools, especially those that operate at the system level.*
LOCKHEED TEAM RESPONSE: The RASSP Design Environment will not support one set of integrated tools that comprise a point solution, but, rather, will allow any commercial tool needed by the RASSP user to be encapsulated. Currently, five encapsulation methods are used: DYNAMIC LIBRARY ENCAPSULATION, ENCAPSULATION SCRIPTS, DIRECTORY MONITORING, VENDOR-SUPPLIED HOOKS, and ENTERPRISE FRAMEWORK.

7. *High-level modeling of the design and development process is a serious endeavor. The emphasis that this team has placed on this problem reflects an understanding of this. An experimental methodology is required for developing and maintaining the models. Is there a methodology to capture the "skunk works" experience in order to be able to build up the formal models?*

LOCKHEED TEAM RESPONSE: A formal methodology has not been establishing because the appropriate level at which to model the process has not been decided. Too detailed a model will hinder capturing relevant information and will not accurately reflect the process in most cases. Too coarse a model will not be useful. Also, one goal is to use the workflow manager metrics collection facility to collect data automatically.

8. *It was not clear that logistics support of end item designs or products was given appropriate emphasis, given its dominance in the cost of many government programs.*

LOCKHEED TEAM RESPONSE: Emphasis on the acquisition of logistic support tools and their integration into the RDE has been increased. Contacts are being made and tools acquired from each service and from a number of logistic tool vendors.

9. *Better interaction with technology base contractors is needed. We hope that this becomes an opportunity to develop creative approaches to such interaction and interchange, perhaps by integrating the technology base participants in the electronic interchange.*

LOCKHEED TEAM RESPONSE: This is a recognized weakness in Lockheed Sanders' program and a new process was defined to correct it. With the announcement of the new Industrial Base Tech Base contracts (October, 1994), each Tech Base contractor (University and Industrial) will have a single point of contact within Lockheed Sanders' RASSP team. This individual will monitor the progress of the Tech Base contractor and define how the contractor's work can be used in the RASSP process